10-1-2003

Identification and prediction of nonlinear dynamics

Andrew Dick

IDENTIFICATION AND PREDICTION OF NONLINEAR DYNAMICS

Andrew J. Dick

This is a thesis submitted in partial fulfillment of the requirements for the degree of **Master of Science** in **Mechanical Engineering**

Department of Mechanical Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology

Approved by:

Professor_____
Dr. Josef S. Török (Thesis Advisor)

Professor_____
Dr. Stephen Boedo

Professor_____
Dr. Daniel B. Phillips

Professor_____
Dr. Edward C. Hensel (Department Head)

October 2003

Permission Grant

I, Andrew J. Dick, hereby grant permission to the Wallace Memorial Library of the Rochester Institute of Technology to reproduce my thesis in whole or part, as long as no reproductions will be used for commercial use or profit.

Permission granted by:_____
Andrew J. Dick (Author of this thesis)

Date: 10/31/03

ACKNOWLEDGMENTS

I would like to thank my parents for supporting my during my time at R.I.T.

I would also like to thank Dr. Török for all that he has taught me both in class and out. I would like to thank him for his direction and guidance that he gave me in my pursuit of knowledge. I would especially like to thank Dr. Török for all the help that he has provided me by editing my thesis and making the necessary preparations to defend it.

I would also like to thank the four students that I was fortunate enough to work with in Senior Design, Jeremy Redlecki, James Streeter, Joe Houtz, and Ashley Rice. Thanks for sharing my interest in Nonlinear Dynamics and helping provide the means for future students to have the same opportunity.

Thanks to the faculty of the Department of Mechanical Engineering and Kate Gleason College of Engineering of putting forth the extra effort to provide an excellent learning environment for the students. I would also like to thank the staff of the Mechanical Engineering Department and the Kate Gleason College of Engineering for keeping the learning environment functioning.

ABSTRACT

Chaos theory and associated analyses are being applied to a growing number of disciplines. Studies of biological and ecological systems have shown the widest application of chaotic analyses thus far. When studying these systems, it is often only possible to measure a subset of the system's many variables. To effectively perform a number of the analyses required to study a chaotic system, it is necessary to identify a complete strange attractor for the system. Consequently, it is necessary to reconstruct the system's strange attractor from the available data. Many different methods exist for reconstructing strange attractors, but the effectiveness of each of these methods has not been studied and compared. This investigation examines the effectiveness of various reconstruction methods used to preserve the fractal structure of the attractor and the exponential divergence of nearby trajectories in an effort to determine the optimal method for reconstructing strange attractors. With an optimal method to reconstruct strange attractors for chaotic physical systems, engineers and scientists can more successfully characterize a nonlinear system and apply methods to predict its future behavior.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT.)

LIST OF FIGURES

LIST OF FIGURES (CONT.)

INTRODUCTION

Nonlinear system theory and associated analyses are being applied to a growing number of disciplines. These currently include fields such as engineering, chemistry, economics, and physics. The concepts and analyses pertaining to nonlinear systems exhibiting chaotic behavior are even being applied to various ecological systems as well as biological ones. Studies are currently underway to gain a better understanding of the nature of meteorological phenomena, including ozone concentrations by studying fractal dimensions[19]. Certain characteristics of chaotic attractors are also being used to examine biological systems ranging from extensive study of heart rate variability[34,35,36,49] to the examination of brain patterns of schizophrenic patients[15].

The majority of the chaotic physical systems to which the theories of nonlinear dynamics are being applied contain more variables than can easily be monitored. In many of the systems, a number of the variables are difficult if not impossible to measure. Other systems are too complex and contain such a large number of variables that observing all of them would be prohibitive. Initially this presents a problem to those studying nonlinear systems because for most analyses it is absolutely necessary to obtain a complete strange attractor for the system.

To solve this problem, a number of different methods have been developed to reconstruct a system's strange attractor when only a limited portion of information from the system's variables is available. While some methods have more theoretical basis than others, almost every method has exhibited limited success in reconstructing various chaotic attractors. However, many of the numerical studies conducted with these methods were done using only a few sets of data. On a global scale, there is no definitive means to determine which of the reconstructive methods will exhibit the best

performance and produce the most accurate reconstruction of the system's strange attractor.

The goal of this study is to examine a number of the different methods used to reconstruct attractors and examine how each one performs. Two properties are used to determine how well each of the methods reproduce the attractor. The first property is the fractal dimension of the attractor. This value is related to the fractal structure of the attractor and how the data points are distributed throughout the state space. The other property examined is the exponential divergence of nearby trajectories within the attractor. As this is one of the defining characteristics of chaotic behavior, it is very desirable to have the rate of divergence maintained as the attractor is reconstructed. This property is measured in the form of Lyapunov exponents.

In this study, methods will be selected to measure these key properties of the attractors. To measure the fractal dimension, a number of different methods exist. These include the Capacity Dimension, Information Dimension, and Correlation Dimension. Of these three methods, the most successful one will be chosen to measure the fractal dimension of the original attractors and the reconstructed attractors for comparison. Because each of the methods can produce an accurate measure of the fractal dimension, the most efficient method, determined by the degree of accuracy per unit of time, will be selected.

This thesis presents a comprehensive look at all the theory and methods used within this study. From the basics of nonlinear dynamics to selecting the optimal reconstruction method, this thesis presents each step of the process. After defining nonlinear systems and how they differ from linear systems, the thesis will explore concepts including strange attractors, fractal dimensions, and Lyapunov exponents. Each of the algorithms explored and used within the study will be explained in great depth including their implementation into MATLAB M-files. Finally, this thesis will discuss in detail the analysis done to determine the optimal method for attractor reconstruction.

Once the paramount method for reconstructing strange attractors is selected, it will be applied to data collected from actual physical systems. With the optimal method for attractor reconstruction chosen, identification and analysis of chaotic systems will be significantly enhanced.

1 NONLINEAR DYNAMICS

1.1 GENERAL

In order to make this investigation as complete as possible, key concepts in nonlinear dynamics will be reviewed. Starting from the basic concept of dynamics, this review will progress to more advanced topics including Lyapunov exponents, attractor reconstruction, and determining the dimension of a data set.

Dynamics is the study of change in physical systems. It is a very broad field with applications in a vast array of disciplines. The all-encompassing study of dynamics can be applied to mechanical systems, electrical systems, thermal systems, fluid systems, as well as many more. In addition to being applicable to so many different types of systems, dynamics can also be used to study the behavior of complex systems. **Complex systems** are composed of elements from different types of systems. The study of dynamic systems is often divided into two classifications; linear systems and nonlinear systems.

Based on the nature of linear systems and nonlinear systems, different analysis methods are used. Linear systems can be represented by a linear combination of a finite number of simpler equations. As a result of this property, an analytical solution generally exists for linear systems. A spring-mass-dashpot system is an example of a simple linear system. A diagram of this system is shown as Fig. 1.1.

In this system, the motion of a mass is constrained by a spring and a dashpot. When the mass is initially displaced or has an initial velocity, it will respond with a decaying oscillation. The behavior of this system can be modeled using a basic second-order differential equation. In this equation, m represents the mass, b represents the damping constant of the dashpot, and k represents the spring stiffness. The variable x represents the position of the mass relative to some reference position and the first and

second derivative with respect to time of x represent the velocity of the mass and acceleration of the mass, respectively.

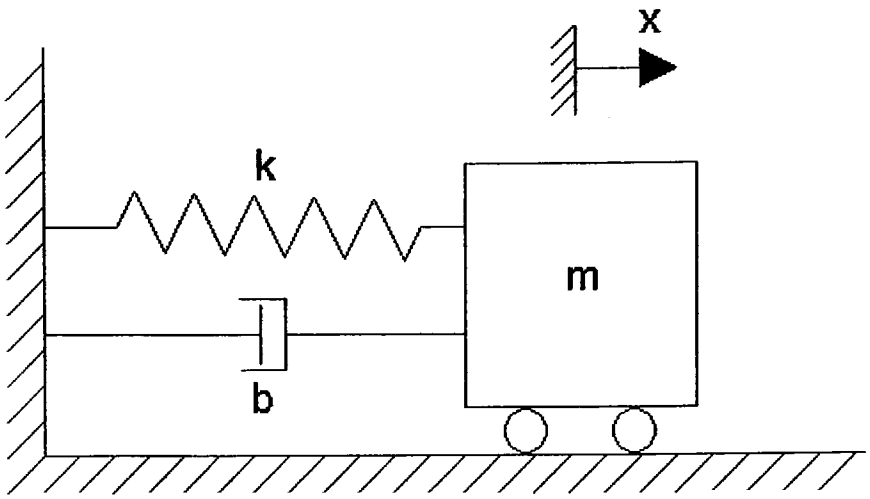


Figure 1.1: Spring-mass-dashpot System

Equation 1.1 shows the second-order linear differential equation that governs the motion of a spring-mass-dashpot system.

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0 \tag{1.1}$$

This differential equation can be converted to two first-order differential equations. Through this simplification, the system's state variables become apparent. Equation 1.2 and Eq. 1.3 are the two first-order differential equations that describe the behavior of the system.

$$\frac{dx_1}{dt} = \frac{dx}{dt} = x_2 \tag{1.2}$$

$$\frac{dx_2}{dt} = \frac{d^2x}{dt^2} = -(bx_2 + kx_1)/m \tag{1.3}$$

The two state variables in this system are the displacement and velocity of the mass. Both of these equations are linear in the state variables. A **nonlinearity** is a state variable raised to a power greater than one or a product of multiple state variables.

Because of the dashpot, the response of this system to any initial displacement or velocity will converge to a stable position. The frequency and rate of decay of the response are functions of the mass, stiffness, and damping constant. These three values, m, k, and b, are the control parameters of the system. **Control parameters** are the variables in the governing equations that are not the state variables. By changing the value of these parameters, the manner in which the system responds to a particular initial condition or input will be altered. When this system is simulated, it is possible to observe how changing the control parameters affects the system's response. Figure 1.2 shows the response of the system for a given set of control parameter values. By slightly increasing the damping constant of the dashpot, the new response shows a greater rate of decay. Because the mass and stiffness remain the same, the frequency at which the system oscillates will remain nearly the same. This new response can be seen as Fig. 1.3.

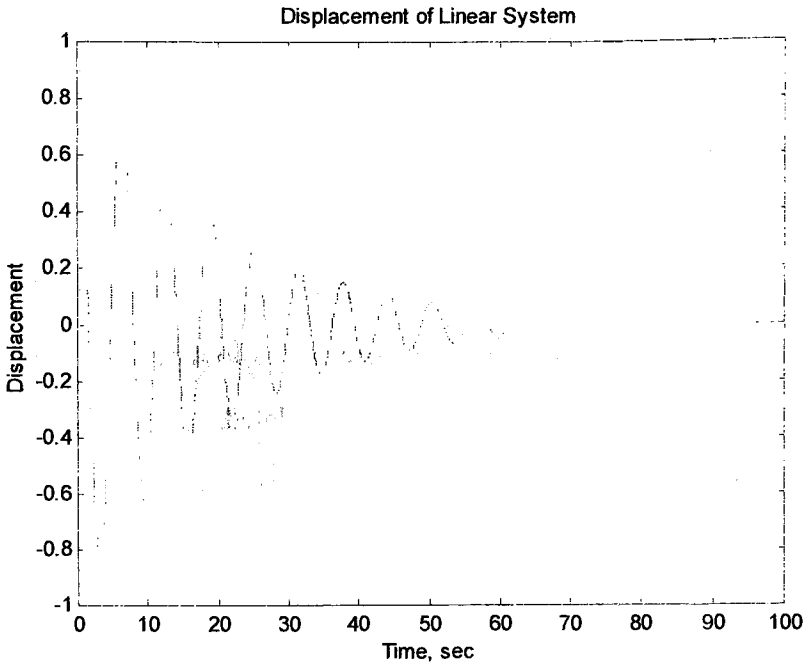


Figure 1.2: Response of Spring-mass-dashpot System for Parameter Values of:

$$m = 1,\ b = 0.1,\ k = 1.$$

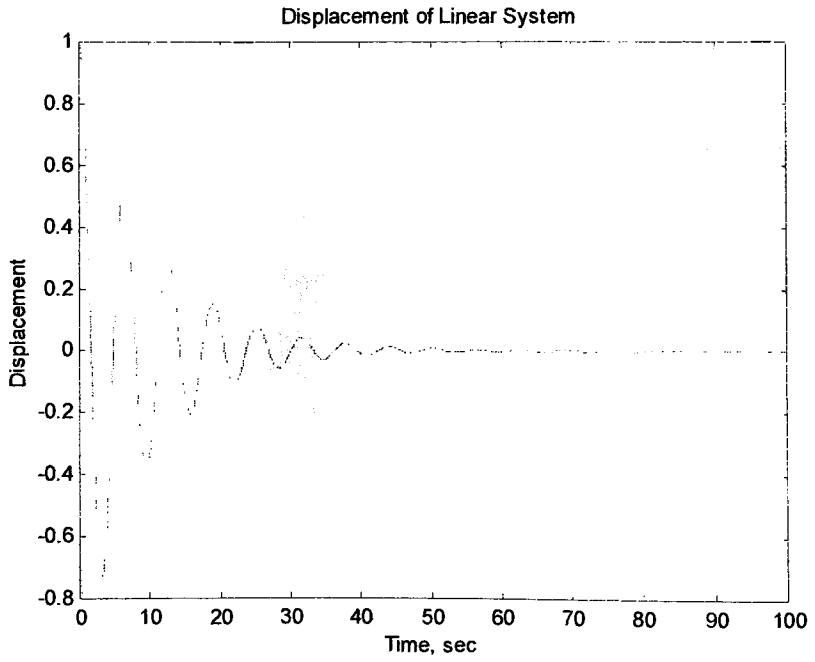


Figure 1.3: Response of Spring-mass-dashpot System for Parameter Values of:

$$m = 1,\ b = 0.2,\ k = 1.$$

Different methods of analyses are necessary when examining nonlinear systems because they are more complex and analytical solutions generally do not exist. The complexity is a result of nonlinearities that exist within the differential equations that govern the dynamics of the system. These nonlinearities generally make it impossible to separate the system into a linear combination of less complex equations. A simple example of a nonlinear system is a damped pendulum. A diagram of this system is displayed as Fig. 1.4.

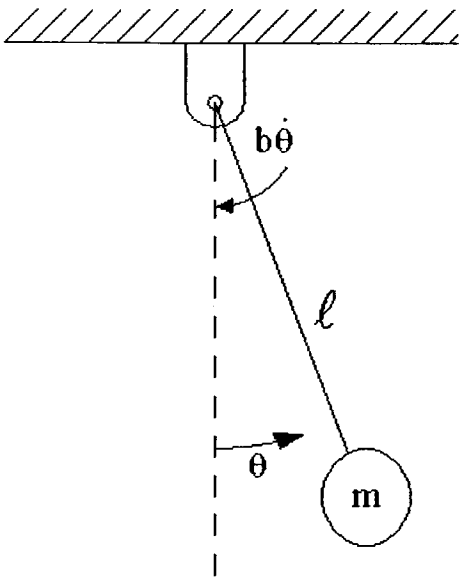


Figure 1.4: Damped Pendulum

When the full range of motion of the pendulum is explored, the differential equation governing the system will include a $sin(\theta)$ term. The equation of motion for a damped pendulum is shown as Eq. 1.4.

$$ml^2\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} + mglsin(\theta) = 0 \tag{1.4}$$

The second-order differential equation for this system can then be separated into two first-order differential equations just as it was done with the linear system. The separation of Eq. 1.4 produces Eq. 1.5 and Eq. 1.6.

$$\frac{d\theta_1}{dt} = \frac{d\theta}{dt} = \theta_2 \tag{1.5}$$

$$\frac{d\theta_2}{dt} = \frac{d^2\theta}{dt^2} = -(b\theta_2 + mglsin(\theta_1))/ml^2 \tag{1.6}$$

The state variables for this system are the angular position and the angular velocity of the pendulum. While the nonlinearity of the system is not as obvious as a squared state variable or product of two state variable, it does exist. Using a Taylor expansion, the nonlinearity of the sine function becomes apparent.

$$sin(\theta) = \sum_{n=1}^{\infty} \frac{\theta^{(2n-1)}(-1)^{(n-1)}}{(2n-1)!} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \dots \tag{1.7}$$

As with the linear system, the variables within the governing equations that are not state variables are the system's control parameters. By adjusting the mass, damping constant, length of the pendulum, or the acceleration due to gravity, it is possible to change the properties of the system's oscillations. Figure 1.5 displays the response of the pendulum to a given set of parameter values. By increasing the value of the damping constant, the rate of decay increases. The response to this increase in the damping ratio can be seen as Fig. 1.6. Unlike linear systems, the values of the control parameters can play a much more important role. When the parameters are modified sufficiently, it is possible to change the dynamics of many nonlinear systems. For a certain set of control parameters, it is even possible for a complex nonlinear system to exhibit chaotic behavior.

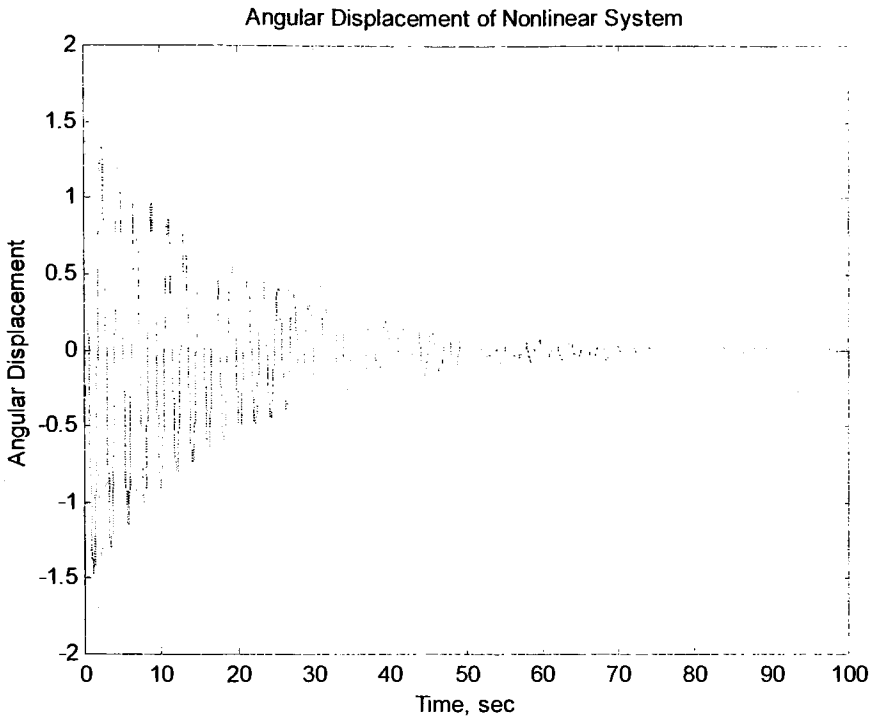


Figure 1.5: Response of Damped Pendulum System for Parameter Values of: $m = 1,\ l = 1, b = 0.1$.

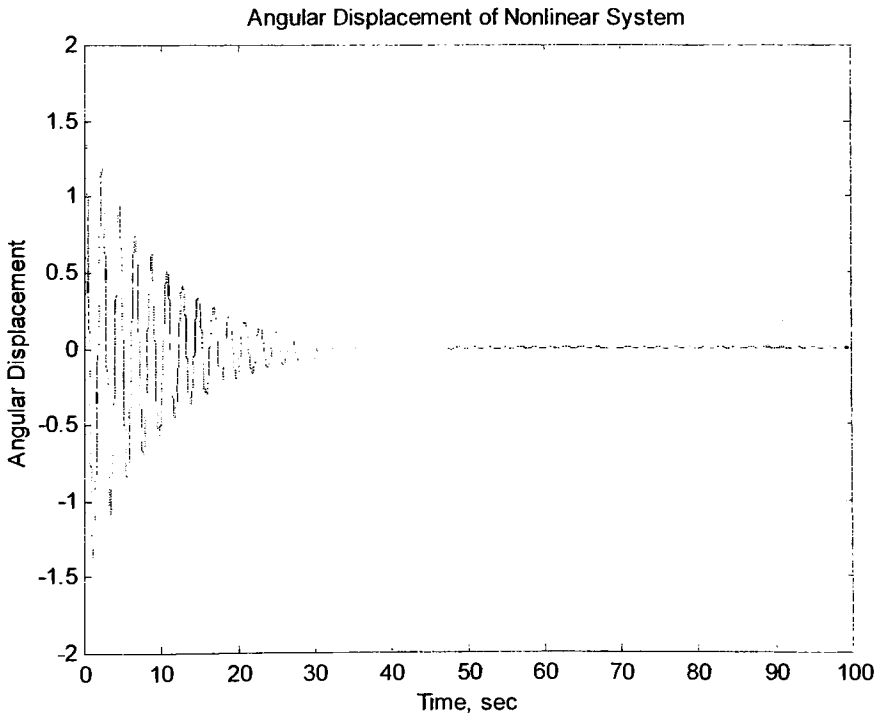


Figure 1.6: Response of Damped Pendulum System for Parameter Values of: $m = 1,\ l = 1, b = 0.2$.

1.2 CHAOS

Chaos is long-term, aperiodic behavior exhibited by a deterministic system that has a sensitive dependence on initial conditions[41]. When observing the time series of a state variable from some chaotic systems, it may initially appear to be periodic over a very large duration, but further investigation will show that the time series is **aperiodic,** meaning that it will never repeat itself. The aperiodic behavior of a time series can be verified using a Fourier analysis. The frequency spectrum produced will contain no distinct frequencies and more closely resemble the broad-band appearance of noise. Figure 1.7 shows the frequency spectrum of a chaotic signal. Figure 1.8 displays a simple periodic signal and an example of a chaotic signal can be seen in Fig. 1.9, both on page 11.

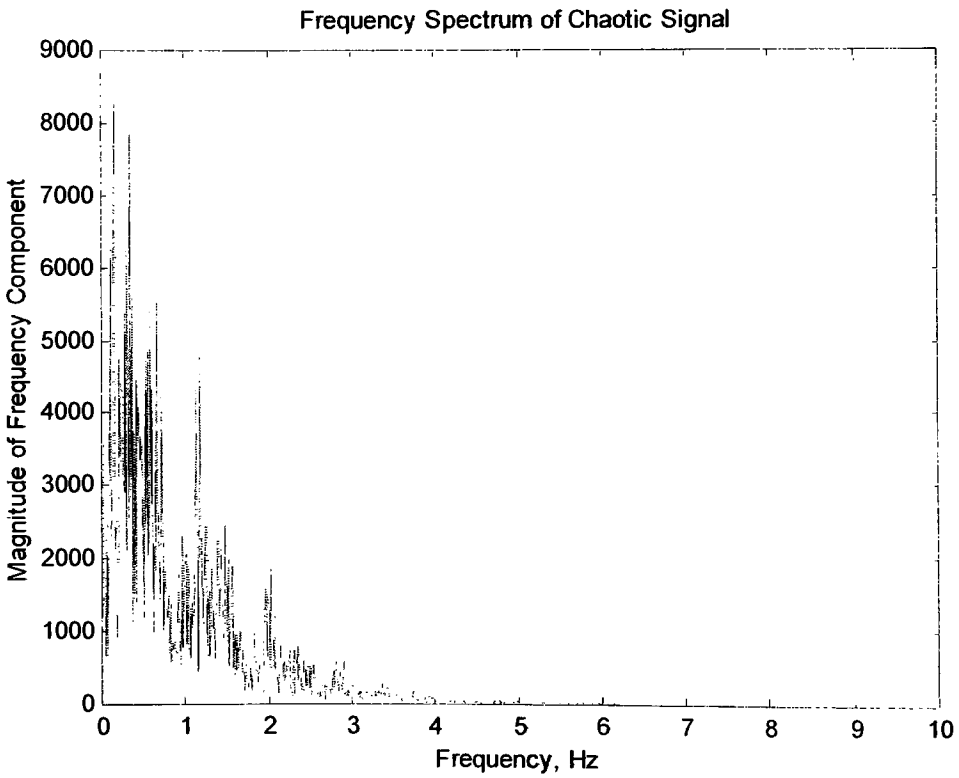


Figure 1.7: Frequency Spectrum of Chaotic Signal

The irregular behavior displayed by chaotic signals is not caused by external noise as one might initially think. This characteristic of the signal is actually caused by

the dynamics of the deterministic system that produced the signal. A **deterministic** system is one that is governed by a well-defined system of equations. For a particular set of state variable values, the system will proceed to exactly on succeeding set of values. In addition to this property, once the trajectory of a chaotic system has passed through one set of state variable values, it will never repeat these exact values. This will be much easier to observe after the concept of state space has been introduced.

One of the most notable characteristics of chaotic behavior is that long-term prediction is extremely difficult, if not impossible. Because the system does not converge to some predictable behavior, precise knowledge of the system's inputs and initial conditions are very important. Due to the nature of a chaotic system, the difference between two time series that start very close together will grow exponentially. As a result, the slightest difference in the initial conditions will cause the system to behave in a completely different manner. Increasing the measurement precision of the initial conditions by many orders of magnitude will only increase the length of time that the behavior of the system can be accurately predicted by a small amount[41,51].

When trying to predict chaotic behavior, the desired precision of the prediction must be determined. This value, δ_{max}, is the maximum separation allowable between the prediction and the actual behavior of the system. Using a relation that will be further explained in Chapter 2, we are able to determine the length of time that the error in the prediction will remain less than the maximum separation. In this formula, λ is the rate of exponential growth or decay of nearby trajectories and $\delta(0)$ is the initial separation between trajectories. These values, called Lyapunov exponents, will be discussed in Chapter Two. The following equations show how the length of time that the chaotic system can be predicted is very strongly dependent on the precision of the measurement of the systems initial conditions[41].

$$t_{pred} \sim O\left(\frac{1}{\lambda} ln\left(\frac{\delta_{max}}{\delta(0)}\right)\right) \tag{1.8}$$

For $\delta_{max} = 10^{-4}$ and $\delta(0) = 10^{-8}$

$$t_{pred} \approx \frac{1}{\lambda} ln\left(\frac{10^{-4}}{10^{-8}}\right) = \frac{1}{\lambda} ln(10^4) = \frac{4*ln(10)}{\lambda} \tag{1.9}$$

For $\delta_{max} = 10^{-4}$ and $\delta(0) = 10^{-14}$

$$t_{pred} \approx \frac{1}{\lambda} ln\left(\frac{10^{-4}}{10^{-14}}\right) = \frac{1}{\lambda} ln(10^{10}) = \frac{10*ln(10)}{\lambda} \tag{1.10}$$

Using Eq. 1.8, the length of time that the chaotic system can be predicted is only two and a half times larger when the precision of the initial condition is increased by a factor of one million. This is how the exponential divergence of nearby trajectories cause chaotic systems to have a sensitive dependence on initial conditions.

Unlike chaotic signals, other types of aperiodic signals exist that are not created by deterministic systems. These **stochastic** signals are produced when the input to a system includes a large amount of random noise. An example of a stochastic signal created using the random number generation function in MATLAB, 'rand', can be seen in Fig. 1.10. While stochastic signals may appear very similar to the chaotic behavior of a nonlinear system, analysis of these signals shows that there is no intrinsic structure. It is very important to be able to distinguish the difference between a stochastic signal and a chaotic signal. It could mean the difference between a complex nonlinear system exhibiting chaotic behavior or a simple linear system with extremely noisy input. Distinguishing the two types of signals can be done by observing the system's behavior using analyses including state space representation of a signal and by calculating the dimension of the data set.

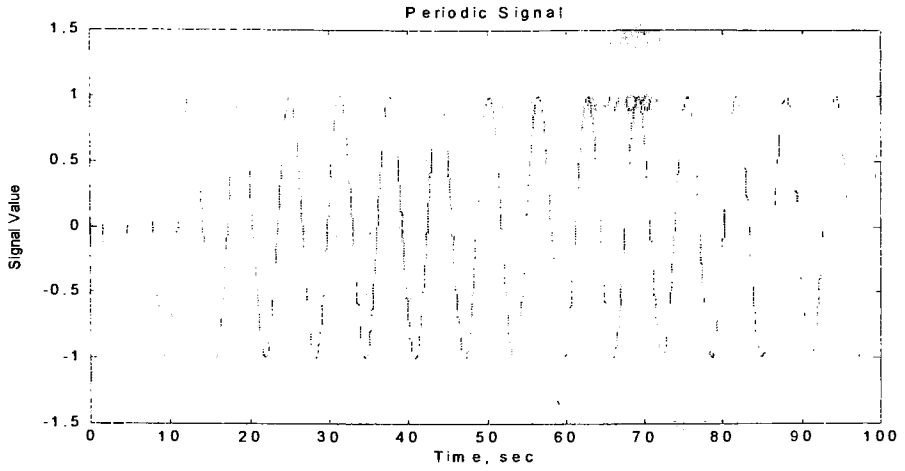


Figure 1.8: Periodic Time Series

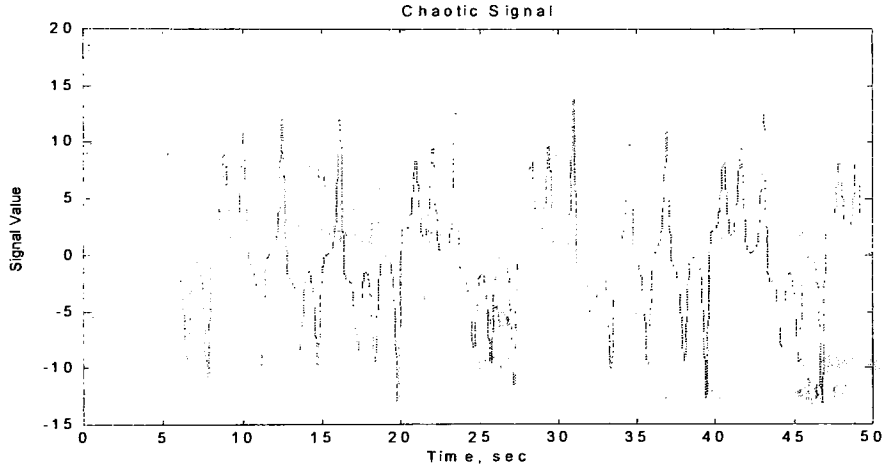


Figure 1.9: Chaotic Time Series

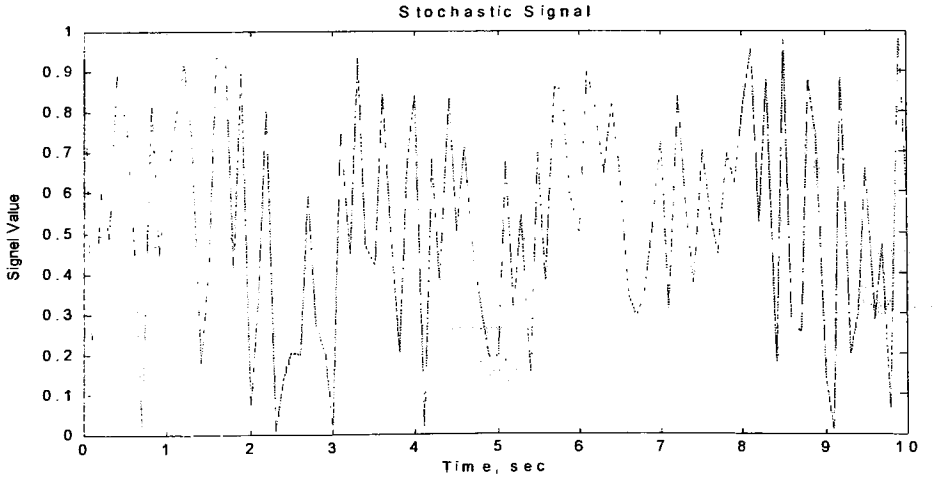


Figure 1.10: Stochastic Time Series

1.3 STATE SPACE

State space, also known as phase space, is a multi-dimensional space created by assigning each of the system's state variables to an orthogonal axis. State space is a common method for observing the behavior of dynamic systems. A system with three state variables can be examined in a three-dimensional vector space or a two-dimensional projection. One system to which this process can be applied is the Lorenz system. The **Lorenz equations** are three first-order differential equations with two nonlinearities. They were developed by Edward Lorenz in 1963 to model atmospheric convection[23]. This system of equations can be seen below.

$$\frac{dx}{dt} = \sigma(y - x) \tag{1.11}$$

$$\frac{dy}{dt} = rx - y - xz \tag{1.12}$$

$$\frac{dz}{dt} = xy - bz \tag{1.13}$$

From these equations the state variables are recognized as x, y, and z. The state space for this system is created by assigning each of these state variables to a perpendicular axis. Figure 1.11 shows the state space of the Lorenz system.

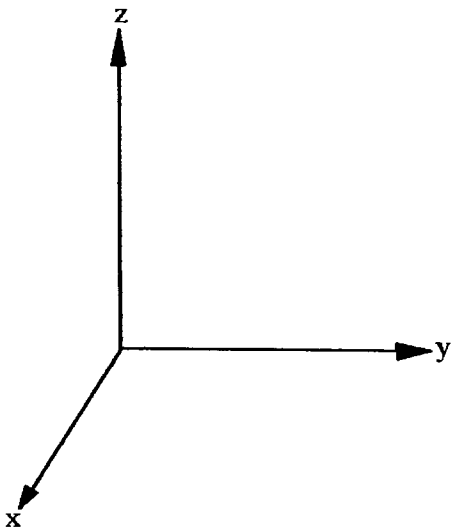


Figure 1.11: Lorenz System State Space

Within the state space, the three time series produced by the system are plotted simultaneously to display a trajectory within the space. A **trajectory** is the evolution of a set of state variable values in time as seen in state space. This method can be applied to both linear and nonlinear systems, but it is especially effective in determining whether a signal is chaotic or stochastic. When the system's behavior is plotted in this multi-dimensional space, the system's dynamics become geometrically visible and it is much easier to observe the inherent structure within the signal.

1.4 ATTRACTORS

Attractors are geometric entities associated with long-term solutions to the equations that govern the dynamics of a system. They are subsets of the state space to which all trajectories within the basin of attraction will converge. The **basin of attraction** is the set of all initial conditions within the state space whose trajectories will converge toward an attractor.

Depending on the behavior of the system, a variety of different solutions can be seen. These include fixed points, limit cycles, and chaotic attractors. A **fixed point** is a point within the state space where the rate of change in the system's state variables is equal to zero. A **limit cycle** is a closed curve on which the change in the state variables will cause the trajectory to remain on the closed curve. Limit cycle solutions can exist only for nonlinear systems.

A fixed point can exist in various states of stability including stable nodes, stable spirals, unstable spirals, and unstable nodes. When a stable fixed point exists in the state space of a system, trajectories within the basin of attraction will converge to a stable position. Figure 1.12 contains two time series of a system as it converges to a stable position. Both the displacement and velocity of the system converge to steady-state values. When this system is examined in a two-dimensional state space, as seen in Figure

1.13, the details of the attractor become clear. For this system, the point attractor is a stable spiral at the origin.

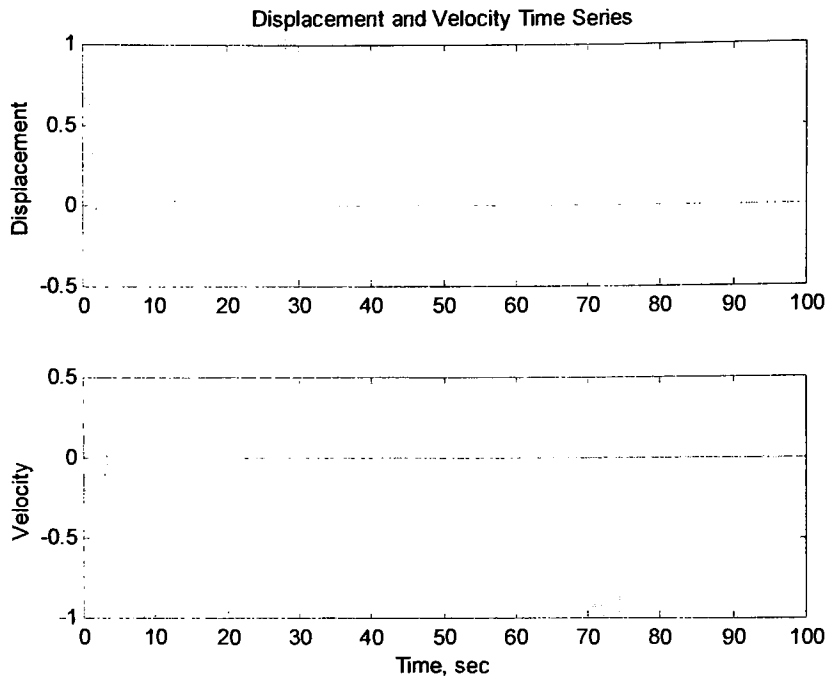


Figure 1.12: Time Series of Displacement and Velocity Approaching Point Attractor

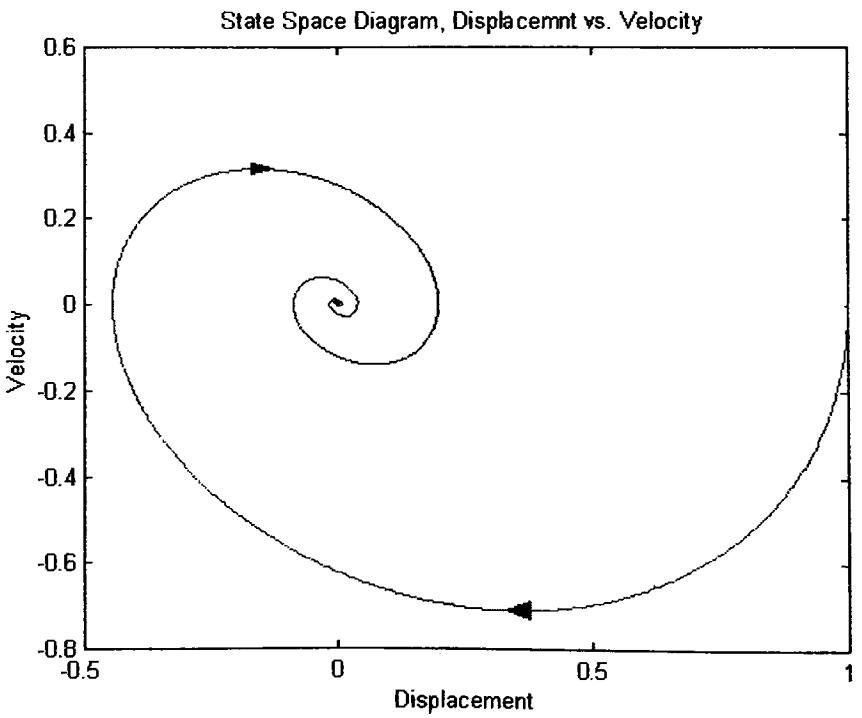


Figure 1.13: System Converging to Point Attractor in Two-Dimensional State Space

Limit cycles can also exist in different states of stability. If a trajectory initially starts exactly on an unstable limit cycle, it will remain on the closed curve but initial conditions infinitesimally off of the limit cycle will cause to trajectory to quickly move away from the closed curve. When a stable limit cycle exists, the behavior of the system will converge to the stable periodic solution. Figure 1.14 shows the time series of a system as it converges to a periodic solution. The periodic behavior can be seen in the displacement time series as well as the velocity time series. The two dimensional state space in Fig. 1.15 shows the two time series plotted against each other. In this diagram, it is possible to see the trajectory converge to the stable limit cycle. The limit cycle attractor in this diagram has been darkened for emphasis.

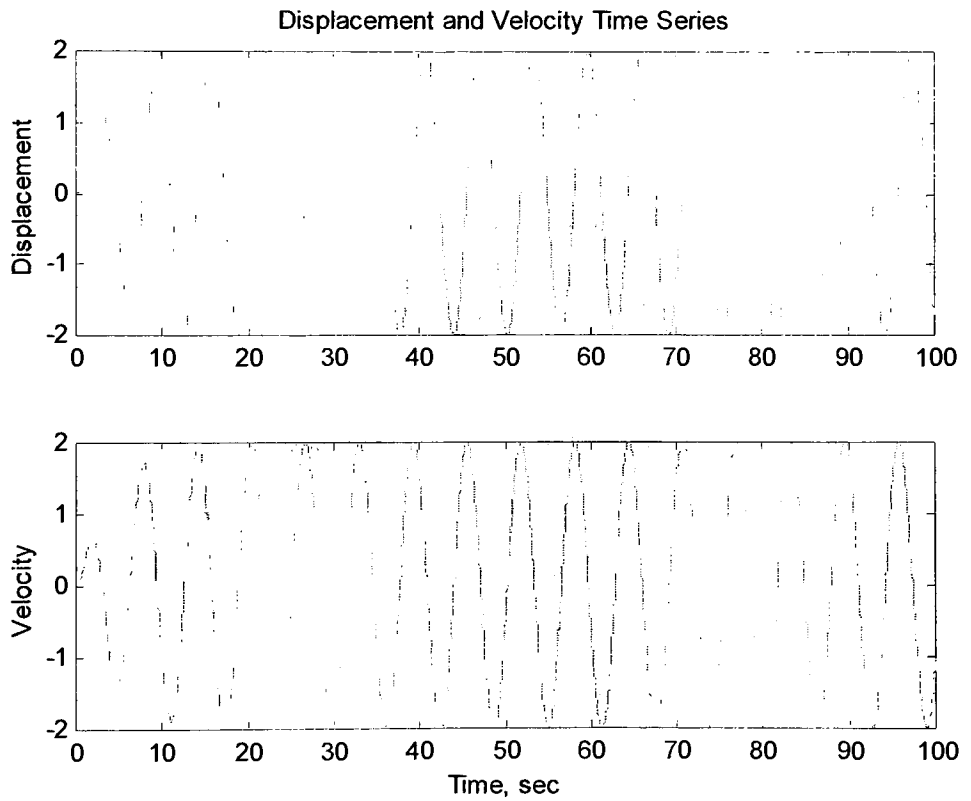


Figure 1.14: Time Series Approaching Limit Cycle Attractor

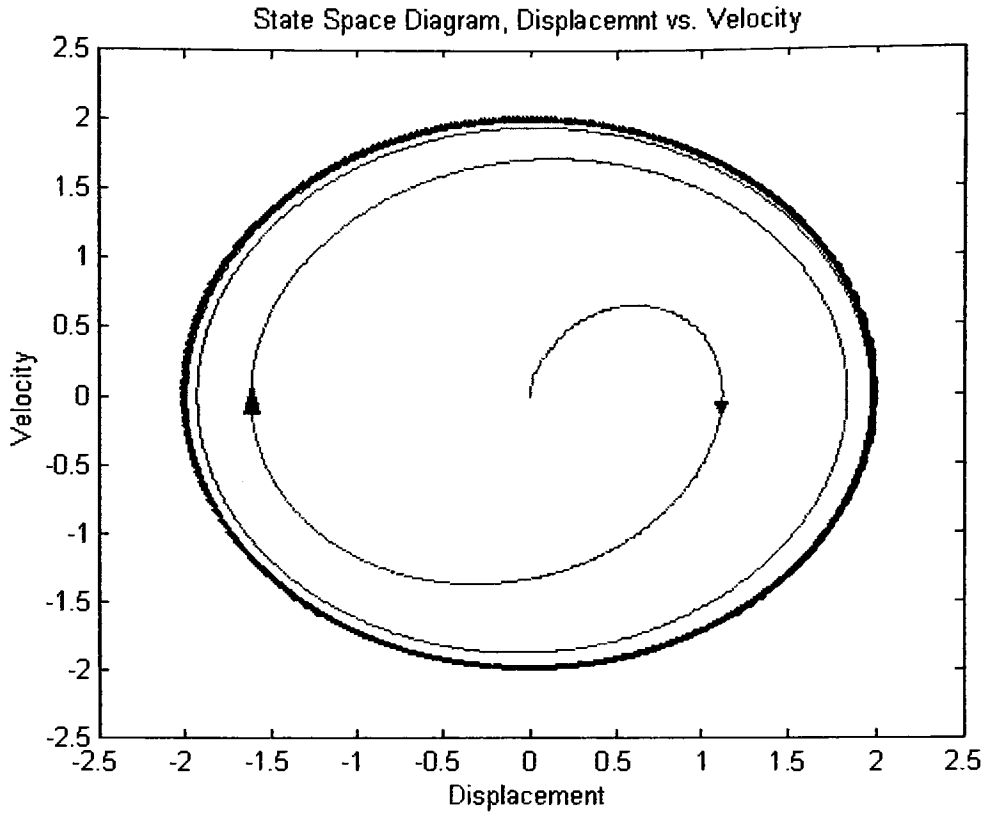


Figure 1.15: Converging to Limit Cycle Attractor in Two-Dimensional State Space

Strange attractors, also called chaotic attractors, are the attractors produced by systems exhibiting chaotic behavior. These attractors are very different from the others. They differ from fixed points and limit cycles with respect to their stability. Unlike the other solutions, a strange attractor can only exist in a stable form. When a chaotic attractor is present, it will draw nearby trajectories into it. Another major difference between strange attractors and the other attractors is the complexity of strange attractors. While trajectories attracted to a point attractor converge to a single point and trajectories attracted to limit cycle converge to a closed loop of points, a chaotic trajectory converges to a specific volume in state space. Because strange attractors exist in the form of a volume within state space, chaos can only be found in systems with at least three degrees of freedom. Once a trajectory converges to this strange attractor, it remains within the volume but does not ever repeat the same path within state space or intersect itself. Two

of the three chaotic time series produced by a Lorenz system are shown in Fig. 1.16. Because the Lorenz system for this set of parameters produces relatively basic chaotic behavior, the time series do not appear as random as the output of a more complex system. However, when the output from the system is examined in state space, the inherent structure of the dynamics becomes visible. Figure 1.17 is a two-dimensional projection of the system's three-dimensional attractor. Although it may appear that the trajectory is intersecting with itself, the true three-dimensional attractor would show that the trajectory does not ever intersect with itself as stated previously.

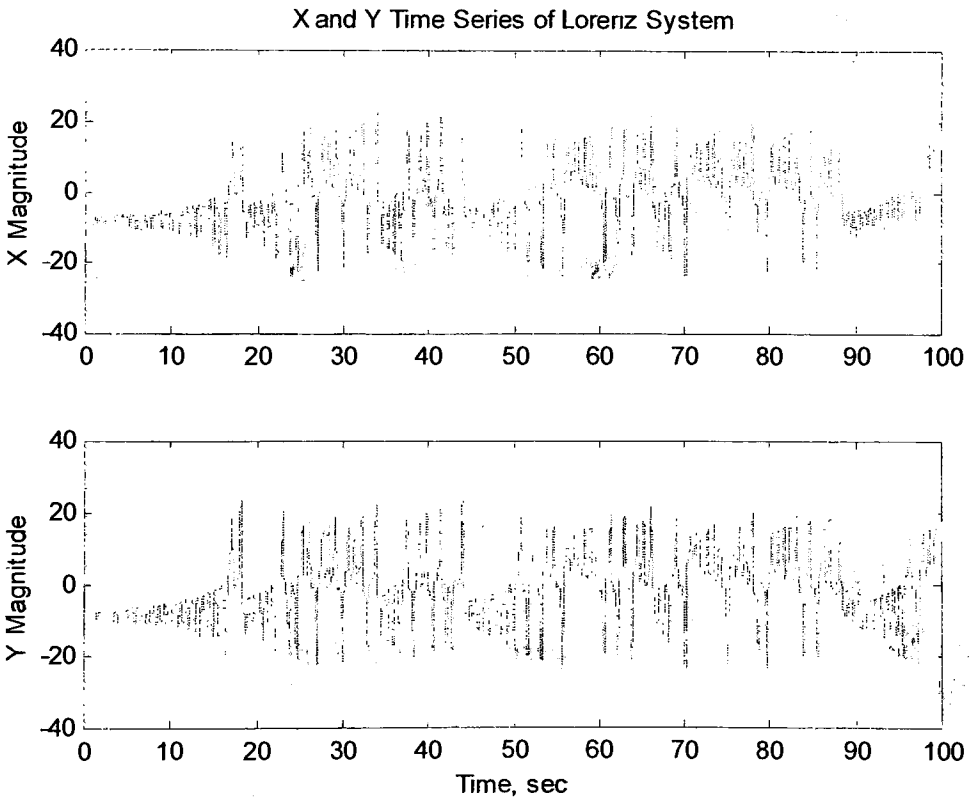


Figure 1.16: Time Series of X and Y Variable of Lorenz Equations

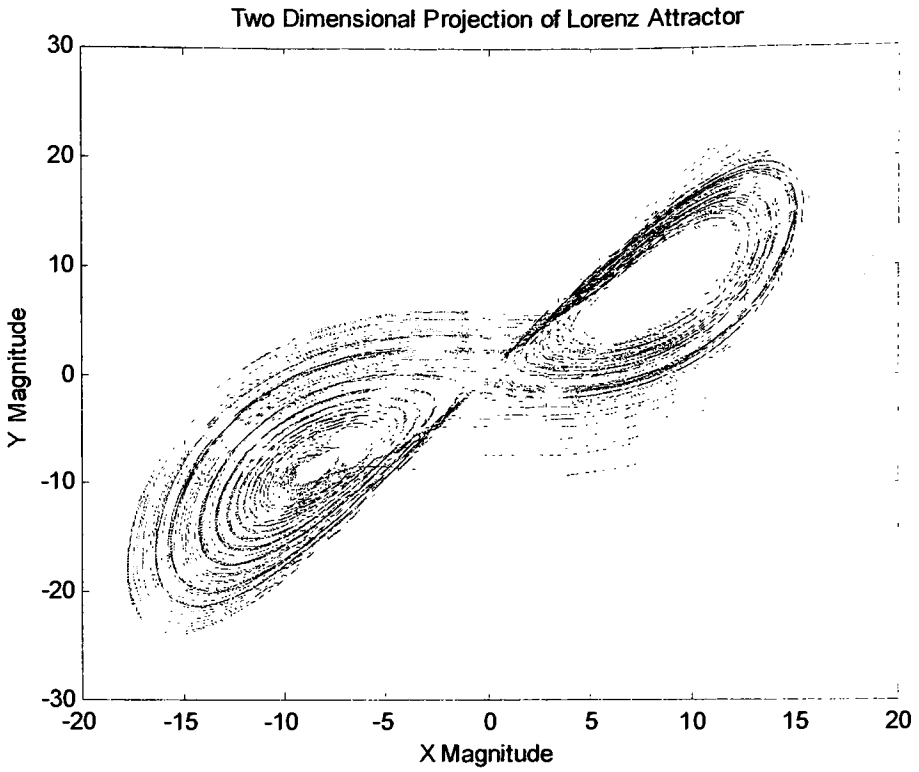


Figure 1.17: Two Dimensional Projection of Lorenz Attractor

Another property of chaotic attractors is that they can only exist when a system is dissipative. In a **dissipative system**, the amount of energy within the system must constantly be changing. The other type of system is a **Hamiltonian** or conservative system, where the amount of energy within the system always remains the same[14]. The change in energy within a system can be determined by evolving a set of initial conditions. By monitoring an area of a two-dimensional system, a volume of a three-dimensional system, or a hypervolume in higher dimensional system, it is possible to classify the system. When an infinitesimal sphere of initial conditions is evolved in a three-dimensional chaotic system, a decrease in the volume of the sphere will verify that the system is dissipative. While the change in the overall volume of the sphere provides information about the system in general, the changes in the size of the sphere along each of its axes provides more detailed information about the strange attractor. The growth or decay of these axes is determined by the attractor's Lyapunov exponents.

2 LYAPUNOV EXPONENTS

2.1 GENERAL

Lyapunov exponents are the average exponential growth or decay rates along orthogonal axes of a trajectory on a chaotic attractor[1,10,14,26,45,51]. They are named after the Russian mathematician A. M. Lyapunov. Lyapunov exponents are very similar to the eigenvalues associated with fixed points in linear systems and can be used to identify the attractor of various systems. The effects of the Lyapunov exponents can be seen when two initial conditions very close together are evolved. The two trajectories start at the initial conditions, $\boldsymbol{x}(t)$ and $\boldsymbol{x}(t) + \delta(t)$ for t equal to zero.

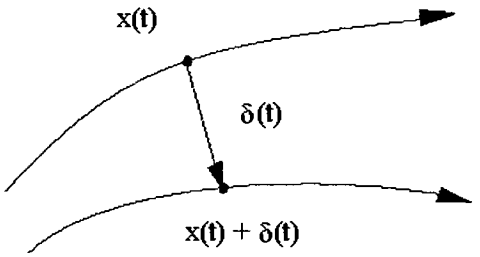


Figure 2.1: Distance between Neighboring Trajectories

As the trajectory evolves, the distance between the two trajectories is determined by the Lyapunov exponent as shown in Eq. 2.1.

$$\delta_i(t) \sim \delta_i(0)e^{\lambda_i t} \tag{2.1}$$

This relation shows how the growth or decay of the value of δ along the ith axis is determined by the value of the ith Lyapunov exponent. The value of this ith Lyapunov exponent can be determined by taking the natural logarithm of the equation and solving for the Lyapunov exponent.

$$ln(\delta_i(t)) \sim \lambda_i t * ln(\delta_i(0)) \tag{2.2}$$

$$\lambda_i \sim \frac{ln(\delta_i(t))}{t} \tag{2.3}$$

Equation 2.3 shows that the exponent is equivalent to the natural logarithm of the separation between trajectories, divided by time. Based on this relationship, a curve is plotted with time as the independent variable and the natural logarithm of the separation as the dependent variable. When this is done, the average slope of a portion of the curve is equal to the corresponding Lyapunov exponent. An example of this graph is shown in Fig. 2.2.

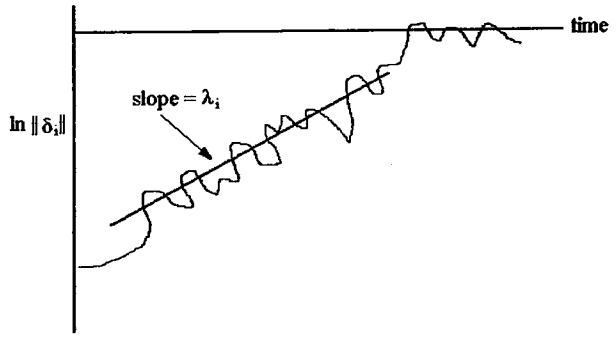


Figure 2.2: Natural Logarithm of Separation versus Time

The nonlinear nature of the curve is due to the variation in the strength of the exponential divergence at different locations on the attractor. The Lyapunov exponent for this direction displays the average of the different strengths across the entire attractor. The separation between trajectories also saturates once it approaches the diameter of the attractor.

When a Lyapunov exponent is positive, it indicates that differences along the associated axis will grow, while a negative Lyapunov exponent indicates that differences along the corresponding axis will shrink. Lyapunov exponents with values very near zero indicate very little change along the corresponding axis. These exponents can be found when analyzing limit cycles, tori, and strange attractors. The rate of growth or decay of the separation of nearby trajectories is also a function of the Lyapunov exponents. The larger the magnitude of a Lyapunov exponent, the greater the growth rate along the axis when it is positive and the greater the decay rate when the exponent is negative. The sum of two Lyapunov exponents will determine the rate of change of an

area of initial conditions and three Lyapunov exponents can be added to determine the change in the associated volume of initial conditions. With the restriction of being in a dissipative system, the sum of the Lyapunov exponents belonging to a chaotic system must be negative. At the same time, at least one of the Lyapunov exponents must be positive to produce the exponential divergence of nearby trajectories that causes the system to have a sensitive dependence on initial conditions. Similar to the constraint on dissipative systems, the sum of all the Lyapunov exponents of a Hamiltonian system must be equal to zero.

The values of the Lyapunov exponents corresponding to a particular system can be used to identify what type of attractor is present in the system. The Lyapunov spectrum for a three-dimensional system would be represented as $(\lambda_1, \lambda_2, \lambda_3)$ where λ_i are the Lyapunov exponents of the system. The standard nomenclature is to label the largest Lyapunov exponent λ_1 and increase the indices of the exponents as their values decrease. Using the Lyapunov spectrum notation, a system with all negative Lyapunov exponents, $(-, -, -)$, would indicate a point attractor within the three-dimensional system. When one of the exponents is zero and the other two are negative, $(0, -, -)$, the system will have a limit cycle attractor. A Lyapunov spectrum consisting of two exponents equal to zero and one negative would produce an attractor in the shape of a torus. Figure 2.3 shows an example of a torus. A chaotic three-dimensional system would be represented by a spectrum containing a positive exponent, $(+, 0, -)$.

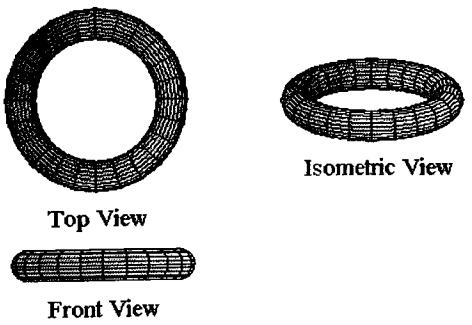


Figure 2.3: A Torus Attractor

2.2 DETERMINATION OF LYAPUNOV EXPONENTS

If the system is known and the differential equations or difference equations are available, it is possible to determine all the Lyapunov exponents of the system[51]. The first step to determine the Lyapunov exponents for a system is to iterate a set of initial conditions until the trajectory of the system is on the strange attractor. Once the trajectory of the system has reached a point in state space that is on the attractor, a number of orthogonal vectors equal to the degrees of freedom of the system must be created. All of the vectors start at the point on the attractor and are of a very small, equal length. It is very important that the length of the vectors be very small because if they are not, the vectors would be subjected to the folding characteristic of the attractor, as well as the stretching. Lyapunov exponents only measure the stretching within the attractor. Once the vectors have been established, the data points must be evolved for a given amount of time. This process is represented in Fig. 2.4.

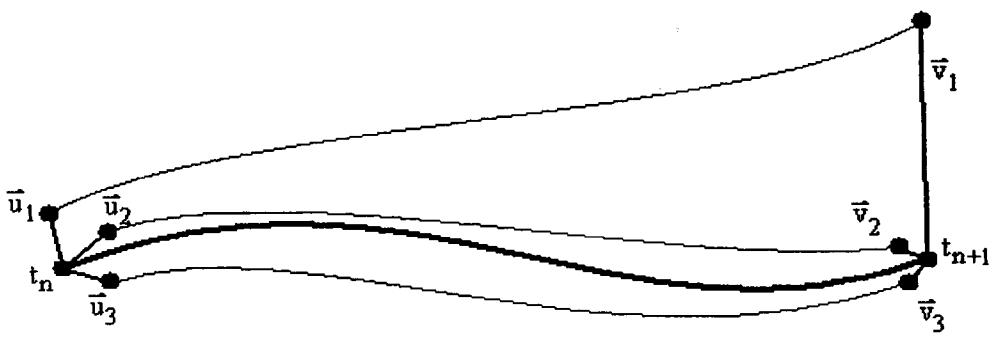


Figure 2.4: Evolution of Vectors Along a Trajectory of a Strange Attractor

In the case shown in Fig. 2.4, one of the vectors increases in size while the other two decrease. Before any further evolution of the points can occur, the three vectors must undergo Gram-Schmidt orthonormalization. This process will reorient the vectors along orthogonal axes to best identify each of the Lyapunov exponents. It will also be necessary to modify the lengths of the vectors to avoid the effects of folding.

The first step in orthonormalizing the vectors is to order them by their length. The vector that gained the most length will be first and the vector that decreased the most in length will be last. With the vectors in order, they are ready to be orthogonalized. This is done by removing the component of the second vector that is in the direction of the first vector. Then the components of the third vector that are in the direction of the first and second vectors must be removed. This process is continued for the rest of the vectors in the case of a higher order system. Equation 2.4, Eq. 2.5 and Eq. 2.6 display the calculations required for a three-dimensional system.

$$\vec{u}_1 = \vec{v}_1 \tag{2.4}$$

$$\vec{u}_2 = \vec{v}_2 - (\hat{e}_1 \cdot \vec{v}_2)\hat{e}_1 \tag{2.5}$$

$$\vec{u}_3 = \vec{v}_3 - (\hat{e}_1 \cdot \vec{v}_3)\hat{e}_1 - (\hat{e}_2 \cdot \vec{v}_3)\hat{e}_2 \tag{2.6}$$

These three equations display the orthogonalization process of three linearly independent vectors, $\vec{v}_1, \vec{v}_2,$ and $\vec{v}_3,$ to produce three orthogonal vectors $\vec{u}_1, \vec{u}_2,$ and $\vec{u}_3$. The vectors $\hat{e}_1$ and $\hat{e}_2$ are unit vectors in the directions of $\vec{u}_1$ and $\vec{u}_2$, respectively. The unit vectors are calculated by dividing the corresponding vector by its magnitude. Now that the vectors have been orthogonalized, they must be normalized. This is done by dividing all three of the orthogonal vectors by their magnitudes and multiplying them by the original vector size. The change in their magnitudes are recorded to be used in the calculation of the exponents. After the Gram-Schmidt orthonormalization process is complete, the points can be evolved further. This cycle is completed multiple times to produce an accurate average of the growth and decay rates across the entire attractor.

As the algorithm is evolving and orthonormalizing the vectors, the amount of growth and decay of the vectors are used to determine the Lyapunov spectrum for the system. The largest Lyapunov exponent is the first to be calculated. The largest Lyapunov exponent be calculated using Eq. 2.7.

$$\lambda_1 = \frac{1}{t_c n}\sum_{j=1}^{n} ln\left[\frac{|\vec{v}(t_{j+1})|}{|\vec{u}(t_j)|}\right] \tag{2.7}$$

In this equation, $|\vec{v}(t_{j+1})|$ represents the magnitude of the vector in the direction showing the most growth after it has been evolved and $|\vec{u}(t_j)|$ is the initial magnitude of this vector. When the algorithm has completed the total number of cycles, the sum is divided by the total amount of time that the vectors have been evolved. In the equation, this value is represented by $t_c n$ where t_c is the time that the vectors are evolved in one cycle and n is the total number of cycles. While the summation of these values are being taken, the area produced by the two largest vectors is also monitored. This area is related to the two largest Lyapunov exponents through Eq. 2.8.

$$A(t) \approx A(0)e^{(\lambda_1+\lambda_2)t} \tag{2.8}$$

The natural logarithm of this equation is taken as it was with the equation for the separation of trajectories. With the size of the area before and after it is evolved on the attractor and the length of time it was evolved, it is possible to determine the sum of the first two Lyapunov exponents. To determine an accurate average across the entire attractor, the areas are evolved many times as shown in Eq. 2.9.

$$\lambda_1 + \lambda_2 = \frac{1}{t_c n}\sum_{j=1}^{n} ln\left[\frac{A'(t_{j+1})}{A(t_j)}\right] \tag{2.9}$$

The value of $A'(t_{j+1})$ in this equation is the product of the magnitudes of the largest vector and the second largest vector after they have been evolved along the strange attractor. The product of the magnitudes of the two largest vectors before their evolution is represented by $A(t_j)$. As with Eq. 2.7, after these values have been summed for a large number of cycles, the value is divided by the total amount of time that the vectors were evolved. Because the area was examined, the value produced is the sum of the two largest Lyapunov exponents. After the largest Lyapunov exponent is calculated,

it can be subtracted from this value to produce the second largest Lyapunov exponent. The third largest Lyapunov exponent, or last Lyapunov exponent in a three-dimensional system, is calculated in the same fashion using the volume created by the three vectors instead of the area produced by two. Equation 2.10 displays the relationship between the first three Lyapunov exponents and a volume element on the attractor. Equation 2.11 shows the formula that will lead to the third Lyapunov exponent.

$$V(t) \approx V(0)e^{(\lambda_1+\lambda_2+\lambda_3)t} \tag{2.10}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = \frac{1}{t_c n}\sum_{j=1}^{n} ln\left[\frac{V'(t_{j+1})}{V(t_j)}\right] \tag{2.11}$$

As with the second Lyapunov exponent, the third can only be determined after the calculation of the first and second. In Eq. 2.11, $V'(t_{j+1})$ is the product of the magnitude of three vectors after their evolution and $V(t_j)$ is the product of the initial magnitude of the three vectors. In the case of a three-dimensional chaotic system, the volume will generally be smaller after the evolution. The product of the three Lyapunov exponents will also be negative indicating a dissipative system. The Lyapunov exponents of higher dimensional systems would be determined in the exact same fashion by continuing the process to examine hypervolumes.

When the system being examined is not known and the governing equations are not available, calculation of the system's Lyapunov exponents becomes more difficult. Determining Lyapunov exponents from an unknown system is done using the same theory as with systems with governing equations. By monitoring two points on the available trajectory as they evolve over time within the data set, it is possible to measure the divergence of the points in a chaotic system. However, because it is no longer possible to chose any point in state space, placement of the vectors are confined to points on an orbit of the trajectory that passes by the subject point. Without the ability to use the Gram-Schmidt orthonormalization process, this process is less accurate than

determining the Lyapunov exponents of a known system and a larger number of cycles will be required. As a result of these restrictions, it is very difficult to determine more than just the largest Lyapunov exponent. The behavior of the system with respect to the largest Lyapunov exponent can be examined, because the largest Lyapunov exponent dominates the behavior of the trajectory while it is on the attractor.

The largest Lyapunov exponent is calculated by selecting a subject point on the attractor and determining the closest point that is not on the same orbit of the trajectory. The distance between the two points is recorded and then the distance between the two points is calculated after following the trajectories for a fixed amount of time. After following the trajectory of the points, the closest point to the new subject point is located. The divergence of two points are followed along the two trajectories and the corresponding distances are once again recorded. This process is repeated many times so that an accurate average of the Lyapunov exponent can be determined. Because the closest point to the subject point is not always in the same location relative to the subject point and the orientation of the points change along the trajectory, there is additional error in these calculations. Figure 2.5 shows how this process occurs.

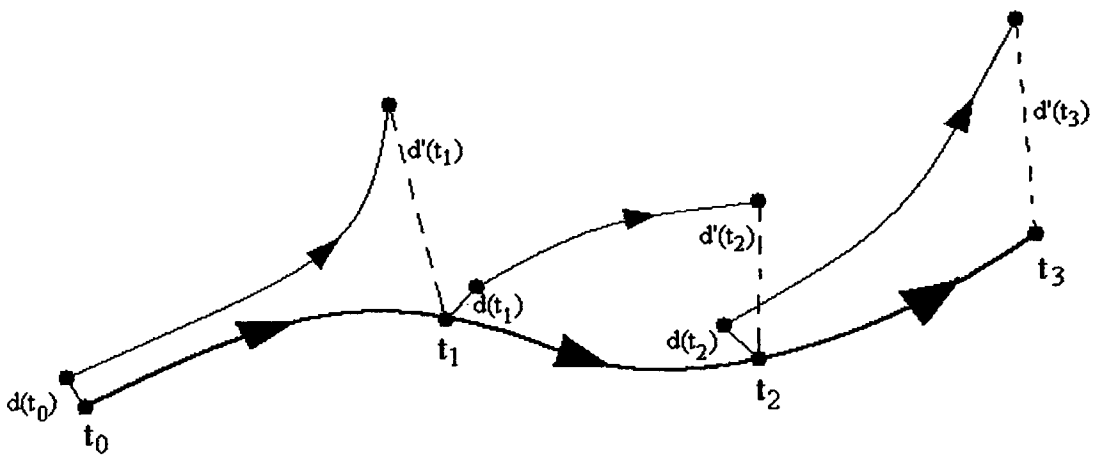


Figure 2.5: Determining a Lyapunov Exponent of an Unknown System

As each cycle is complete and the trajectory of the nearest point is followed, the initial distance and final distance between the data points are used in the calculation of

the system's largest Lyapunov exponent. The formula for calculating this value is the same as Eq. 2.7.

$$\lambda_1 = \frac{1}{t_c n}\sum_{j=1}^{n} ln\left[\frac{d'(t_{j+1})}{d(t_j)}\right] \tag{2.12}$$

Since the largest Lyapunov exponent corresponding to a trajectory on a chaotic attractor is generally much larger than the other exponents, it will dominate the behavior of the trajectory and can easily be determined. Unfortunately, this dominance also makes determining the other Lyapunov exponents much more difficult or impossible.

3 ATTRACTOR RECONSTRUCTION

3.1 GENERAL

With the aid of state space, a dynamic system can be recognized as deterministic and not stochastic. For chaotic systems, the characteristics of the strange attractor can be used to gain information about a data set. Unfortunately when working with actual physical systems, it may not be possible to measure every degree of freedom of the system. Some systems are very complicated making measuring the value associated with some of the state variables very difficult, if not impossible. Other more complicated systems possess a very large number of degrees of freedom and monitoring all of them would be prohibitive. As a result, **attractor reconstruction** has been developed. This process involves various methods to recreate the attractor of a system using only the time series of a single state variable.

While various methods of attractor reconstruction have been studied, most found limited success producing an effective and efficient recreation of the system's attractor. One method consists of recreating the state space vectors using successive derivatives of the available time series[30]. Equation 3.1 shows the formula used to reconstruct vectors from the original time series. This method proved to be impractical because each succeeding derivative amplified any noise or abnormalities within the available time series.

$$\boldsymbol{X}(t) = \left\{ x(t), \frac{dx}{dt}(t), \frac{d^2x}{dt^2}(t), \ldots, \frac{d^{(d_e-1)}x}{dt^{(d_e-1)}}(t) \right\} \tag{3.1}$$

Another method for attractor reconstruction, now the most commonly used method, consists of generating a Delayed Vector Space[14,31,55]. The **Delayed Vector Space Method** recreates the state space vectors using delayed data from the available time series. The concept of reconstructing an attractor using Delayed Vector Space was developed by Floris Takens in the early 1980's to examine the chaotic behavior of

turbulent fluids[42]. After a delay time and embedding dimension are selected, the pseudo-state vectors can be constructed using Eq. 3.2.

$$\boldsymbol{X}(t) = \Big\{x(t), x(t+\tau), \ldots, x(t+(d_e-1)\tau)\Big\} \tag{3.2}$$

In the above definitions, $\boldsymbol{X}$ is the reconstructed vector, x is the time series of the available state vector, τ is the delay time, and d_e is the embedding dimension. The **delay time** is the time between data values from the original time series that will used to construct the delay vectors. The **embedding dimension** is the dimension of the reconstructed state space. This value is the number of components that will exist within the delay vectors. Both the delay time and the embedding dimension must be determined before the system's attractor can be reconstructed and the dynamics of the system can be studied.

3.2 EMBEDDING DIMENSION

Studies have been done to determine how large the embedding dimension must be when reconstructing an attractor using information gained from a time series. Some suggest that the embedding dimension must be at least one plus two times the degrees of freedom of the system, while others suggest that the embedding dimension need only be greater than the number of degrees of freedom of the system[14,24,37,55,56]. Since the number of degrees of freedom of a system is not always known, a more accurate method has also been developed to determine the necessary embedding dimension for a given set of data. This method, known as the False Nearest Neighbors method, examines how concentrated the data points are within state space as the embedding dimension is increased[1,6,21,32]. The concentration is calculated by determining the average number of points within a given radius of all the points in the data set. This value is calculated using the function displayed as Eq. 3.3.

$$FNN(d_e) = \frac{1}{N}\sum_{i}^{N}\sum_{j}^{N} H(R - ||x_i - x_j||) \tag{3.3}$$

In this formula, FNN represents the average number of nearest neighbors within a distance of R of all the points in the data set. When this value is calculated over a range of d_e values, it is possible to plot FNN against d_e and select the appropriate embedding dimension. The function $H(\bullet)$ is the Heaviside function. When the value within the Heaviside function is greater than or equal to zero, the function is equal to one. When the value is negative, the function is equal to zero. The distance between points is calculated in a Euclidean manner. This Euclidean distance can be determined using the formula in Eq. 3.4.

$$||x_i - x_j|| = \sqrt[2]{\sum_{n=1}^{d_e}(x_{i,n} - x_{j,n})^2} \tag{3.4}$$

Once the average number of points within the specified radius of all points converge to a value, the appropriate embedding dimension is known. Because the number of calculations required for this method is proportional to the length of the data set squared, very large data sets will require a considerable amount of time to determine the optimal embedding dimension.

The basis of this method is that when the embedding dimension is too small, the data points are being projected into a lower dimensional state space. As a result the data points will be near each other due to the projection and not the dynamics of the system. The data points that are within a given proximity of the point being examined due to the projection are called false nearest neighbors. The data points around the subject point due to the actual dynamics of the system are known as true nearest neighbors. When the embedding dimension is large enough to eliminate this situation, only true nearest neighbors will be around the subject point.

When examining a finite amount of discrete data collected from an actual physical system, there are also other types of false nearest neighbors that can exist. These false nearest neighbors are a result of the data acquisition and not the process of embedding the data set. One of these types of false nearest neighbors are produced as a result of the rate at which the data is sampled. When the data is sampled with a very small sampling time, additional false nearest neighbors will be present when the embedding process is done. This problem can be solved by resampling the data before calculating the embedding dimension. Another type of false nearest neighbors that can develop during the collection of the data are caused by the rounding and truncation due to the data acquisition equipment. The situation is dependent on the data precision of the hardware and software. These false nearest neighbors can be eliminated by ensuring that the data acquisition equipment has the appropriate precision capabilities. Large amounts of noise can also result in false nearest neighbors that can not be removed by increasing the embedding dimension. This problem can be avoided by taking the necessary actions to reduce the amount of noise in the signal.

A similar approach to determining the required embedding dimension involves calculating the fractal dimension of a data set for a range of embedding dimensions[24,35,37]. As the embedding dimension is increased, the values of the fractal dimension will converge. This method does not require that the embedding dimension be determined before calculating the fractal dimension of the data set but it does require significantly more calculations as the overall length of time to complete the analysis is increased.

3.3 DELAY TIME

Determining the best delay time value is very important when reconstructing vectors using the Delayed Vector Space Method. If the delay time value is too small, the data sets will be very similar. This will result in the data being very concentrated when

plotted into the reconstructed state space and make it very difficult to perform further analysis. This situation is known as *redundance*. A two-dimensional projection of a reconstructed attractor when the delay time value is too small is show in Fig. 3.1. If the delay time value is too large, the two sets of data will become completely uncorrelated and the state space representation will lose much of its structure. This situation is known as *irrelevance*. Figure 3.2 shows the result of a delay time value that is much too large.

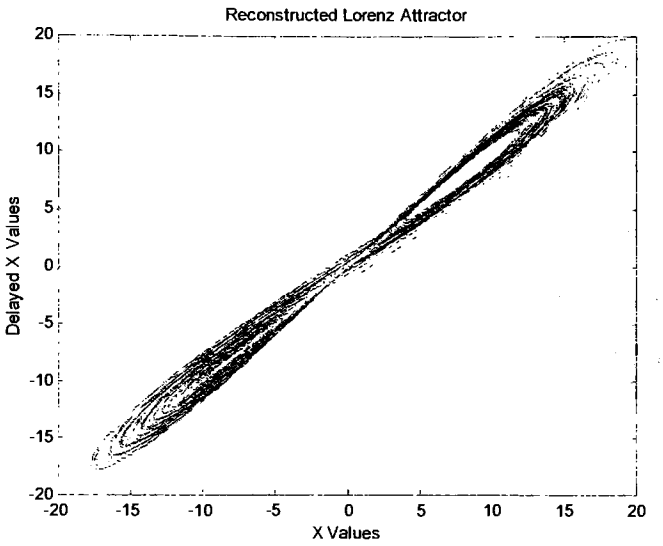


Figure 3.1: Reconstructed Lorenz Attractor with Too Small of a Delay Time

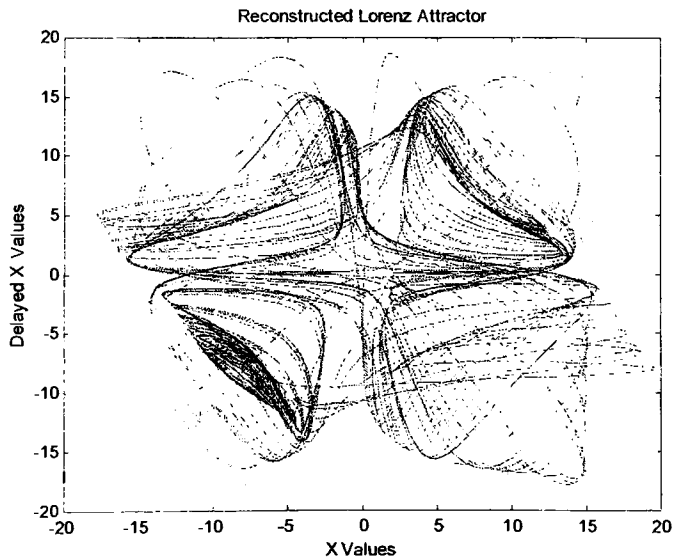


Figure 3.2: Reconstructed Lorenz Attractor with Too Large of a Delay Time

According to Takens, when an infinite amount of noise-free data is available with an infinite precision, any arbitrary delay time can be used to reconstruct the attractor[42]. Since it is not possible to collect a data set that agrees with these requirements, many methods have been developed for the selection of the optimal delay time. The focus of these methods is to identify a relationship between the original time series and a complete range of delayed time series used to create a set of two-dimensional vectors.

3.3.1 VISUAL INSPECTION

The most basic means to determine the optimal time delay value for attractor reconstruction consists of reconstructing the system's attractor for a wide range of delay values. By plotting a two-dimensional or three-dimensional projection of the system's strange attractor, it is possible to look for characteristics of chaos. Across the range of values, the delay time is chosen so that the attractor exhibits the most self-similarity, stretching, and folding. Exponential growth of small distances between nearby points is another characteristic. Figure 3.3 through Figure 3.8 show the progression of attractor reconstruction for a range of delay time values. After examining a set of reconstructed attractors, the delay increments can be decreased and a more precise delay value can be determined. When the delay time value is equal to 0.04 seconds in Fig. 3.3, the data is very near to the 45° bisector. This level of redundance indicates that the delay time value must be greater. As the delay time value approaches 0.20 seconds, the structure of the attractor begins to become distorted signifying the onset of irrelevance. This allows one to increase the precision of the search and examine time delay values between 0.04 and 0.20 seconds. When the original attractor is available as in this case, a comparison between the two can be used to fine tune the choice of the delay time. If a projection of the original attractor is not available when examining experimental data, the attractor should be reconstructed so that it is as distributed as possible within the state space while minimizing the distortion of its structure.

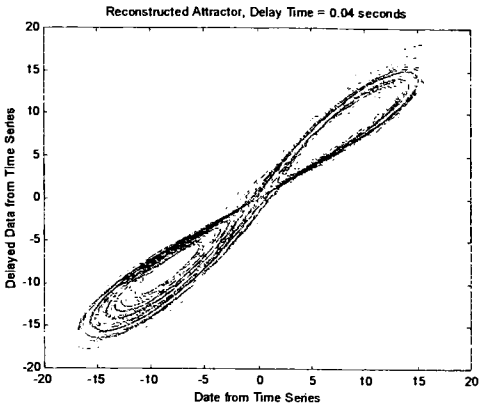


Figure 3.3: $\tau = 0.04$ sec

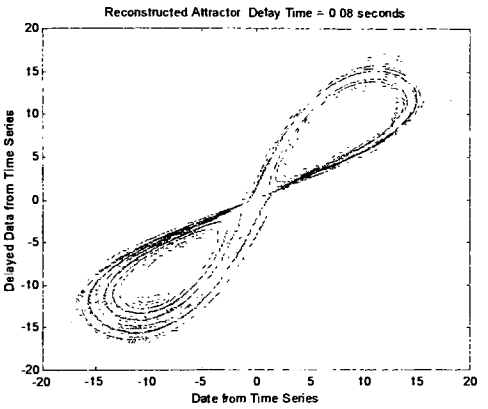


Figure 3.4: $\tau = 0.08$ sec

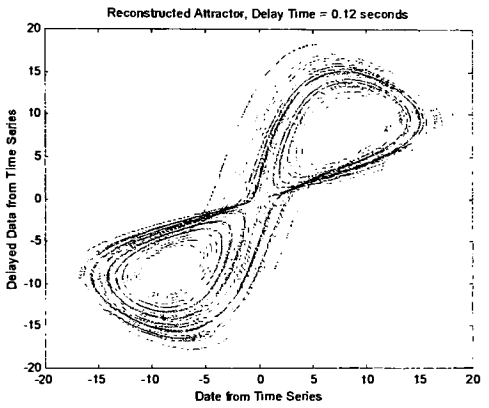


Figure 3.5: $\tau = 0.12$ sec

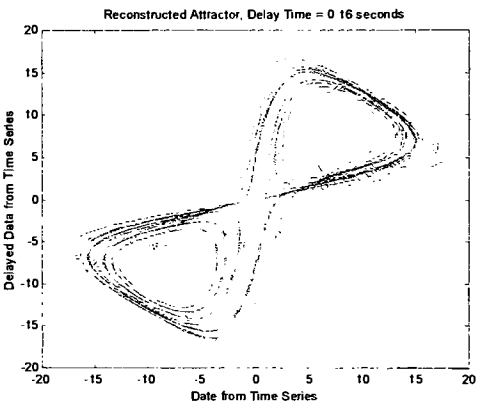


Figure 3.6: $\tau = 0.16$ sec

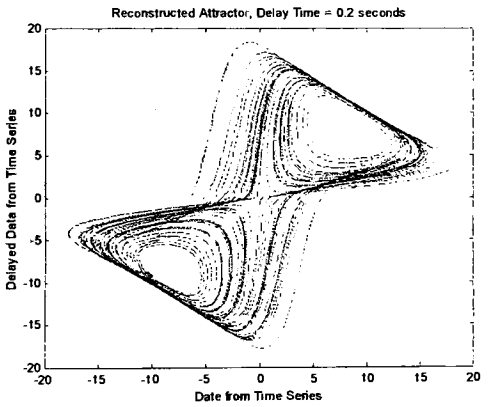


Figure 3.7: $\tau = 0.20$ sec

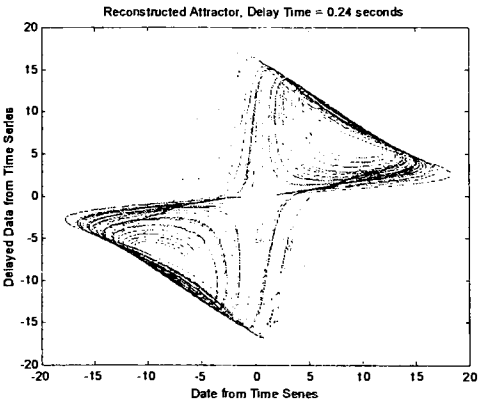


Figure 3.8: $\tau = 0.24$ sec

3.3.2 AUTOCORRELATION METHODS

Another method for determining the optimal delay time value employs the autocorrelation function. The autocorrelation function, defined as Eq. 3.5, provides a quick and easy way to observe a relationship between the original time series and an entire array of delay time values.

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n-l) \tag{3.5}$$

The autocorrelation provides the sum of all the products of the original data values and their corresponding values after they have been shifted. Because actual analyses are completed with a finite amount of data, Eq. 3.5 must be modified so that it is in terms of the length of the data set, N.

$$r_{xx}(l) = \sum_{n=i}^{N-|k|-1} x(n)x(n-l) \tag{3.6}$$

$$\text{where } i = l,\ k = 0, \text{ for } l \geq 0 \text{ and } i = 0,\ k = l \text{ for } l < 0$$

When the autocorrelation function is applied to a set of data, the resulting graph will display the level of correlation between the original data set and the shifted data set for each shift value. When the shift value is equal to zero, the two data sets are identical and the highest level of correlation will exist. Since the autocorrelation of a chaotic data set decays exponentially, with increasing delay time values, the value of the autocorrelation at the position of zero delay will be the global maximum. Figure 3.9 shows the autocorrelation of a chaotic time series produced by the Lorenz equations.

To comply with the size requirements of the delay time value and avoid the errors that occur when the value approaches either extreme, it is necessary to examine the portion of the autocorrelation near the delay value of zero. Due to the symmetry of the autocorrelation, it is only necessary to examine the positive delay values. Figure 3.10 displays a closer look at the autocorrelation function near the delay value of zero.

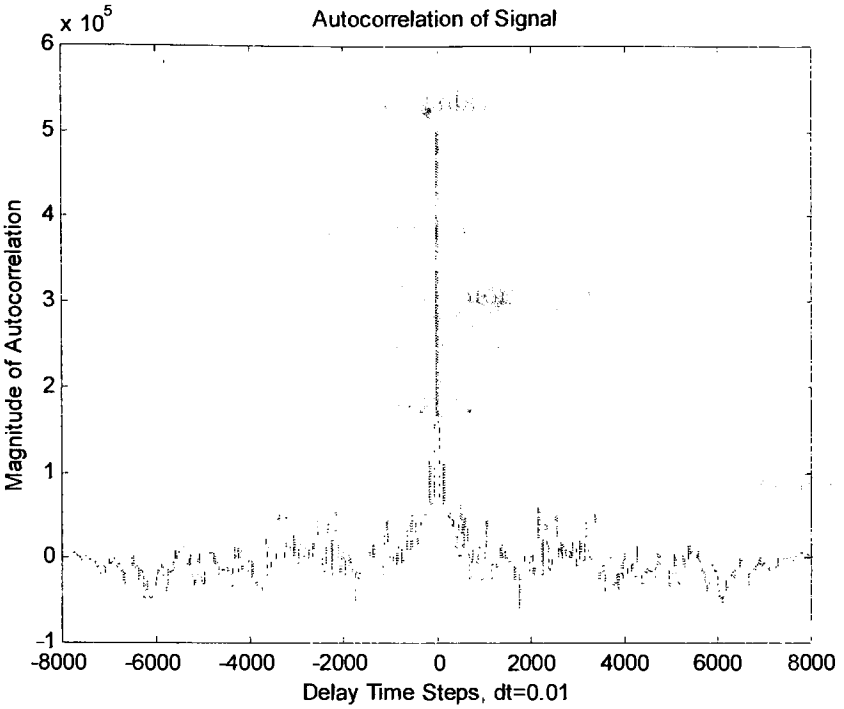


Figure 3.9: Autocorrelation of Chaotic Signal

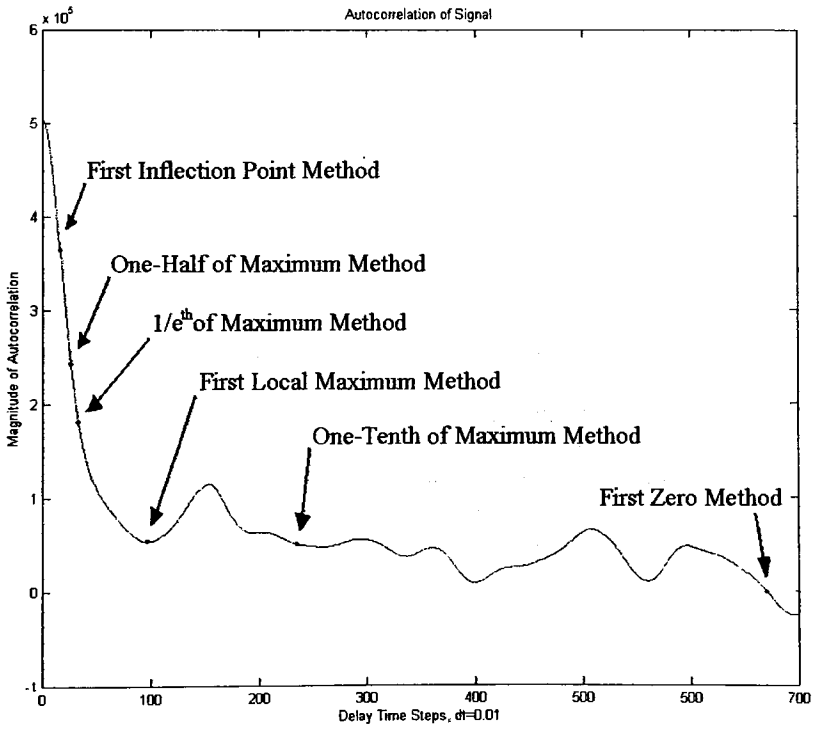


Figure 3.10: Closer View of Autocorrelation of Chaotic Signal

After calculating the autocorrelation of a chaotic time series, there are multiple technics for determining the delay value. To provide a baseline for comparison of these values, the projection of actual attractor is shown in Fig. 3.11. Through trial and error, a reconstruction of the attractor was also produced using a delay time of 0.11 seconds. Figure 3.12 shows this reconstruction.

It was typically considered that when the autocorrelation function was equal to zero, there would not be any correlation between the two data sets and the largest amount of information would be available[1,4,15,24,37]. Figure 3.10 shows the location of the first zero value of the autocorrelation at 6.71 seconds. When the corresponding delay value is determined and the attractor is reconstructed, it becomes apparent that the two data sets have become completely uncorrelated but the reconstructed attractor does not display any structure. The attempt to reconstruct the attractor with this delay time value can be seen in Fig. 3.13. Because the First Zero Method often produces a value much larger than the optimal delay time, when it is employed, the value of the delay time is divided by a number between ten and twenty[24,37,56].

Another method for determining the delay time value involves locating the first local minimum of the autocorrelation function. The first local minimum can easily be calculated using a forward differencing method once the autocorrelation has been obtained. The basis of the First Local Minimum Method is the same as the previous method. When the autocorrelation is at a minimum, there should be a relative minimum amount of correlation between the two data sets[15,24,37,56]. With the erratic structure of the chaotic signal, the autocorrelation will often produce a local minimum before the autocorrelation has a value equal to zero. Figure 3.10 shows the location of the first local minimum at 0.97 seconds. When this value is used to reconstruct the system's strange attractor, the two-dimensional projection shown in Fig 3.14 indicates that this method also produces a delay value much larger than the optimal delay time.

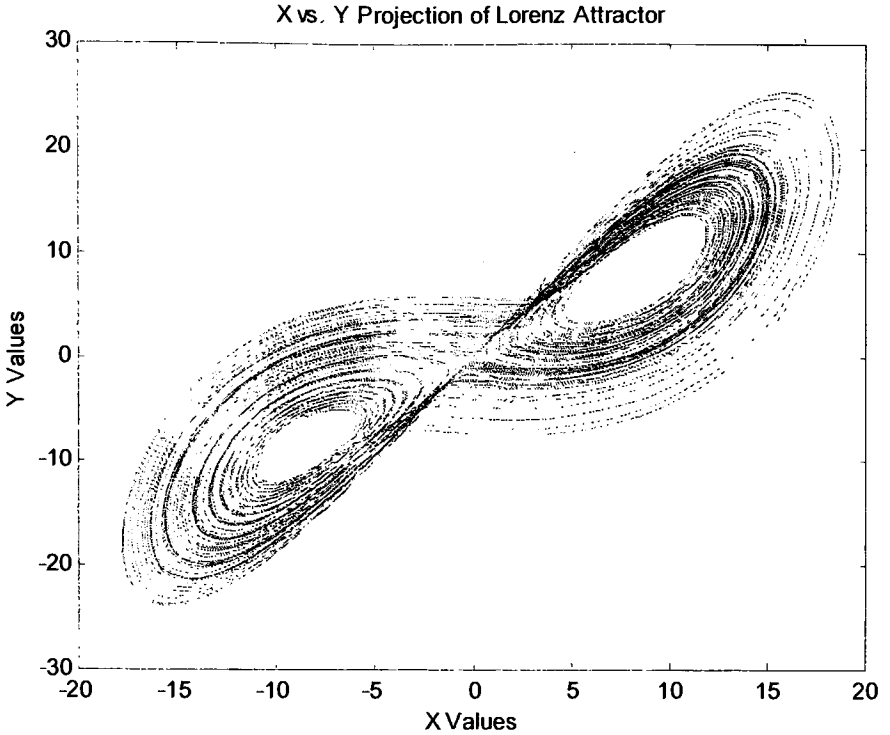


Figure 3.11: X vs. Y Projection of Lorenz Attractor

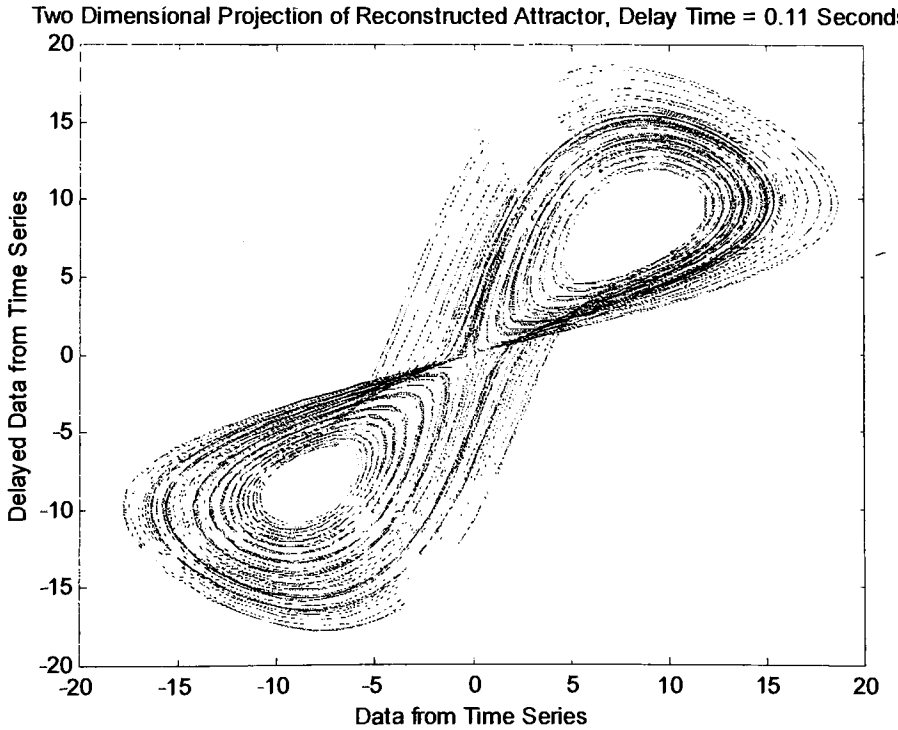


Figure 3.12: Attractor Reconstruction with Delay Time Value of 0.11 Seconds

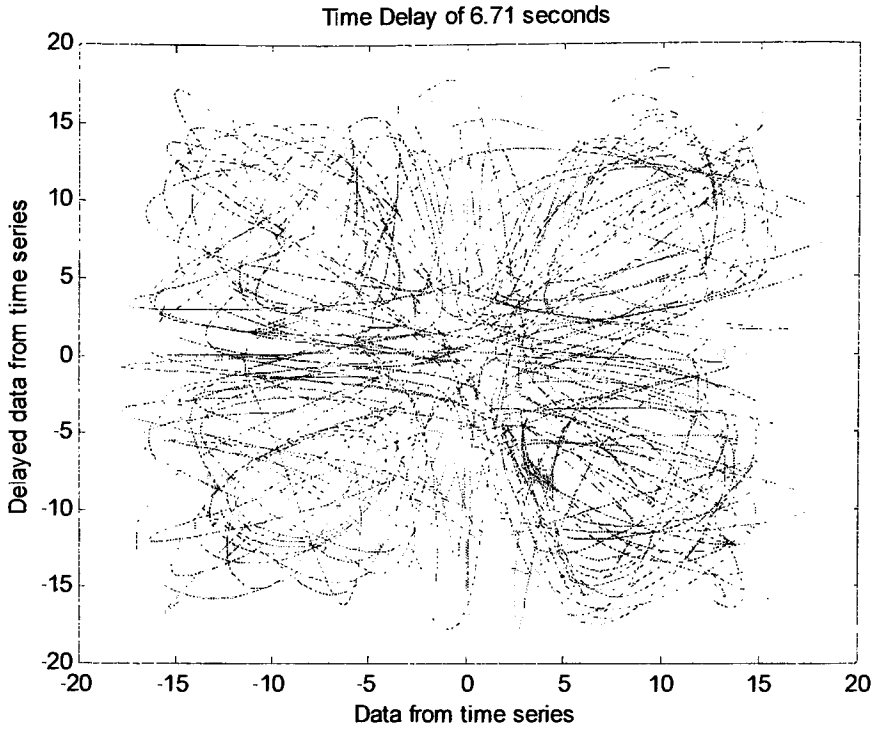


Figure 3.13: Reconstructed Attractor using First Zero Autocorrelation Method

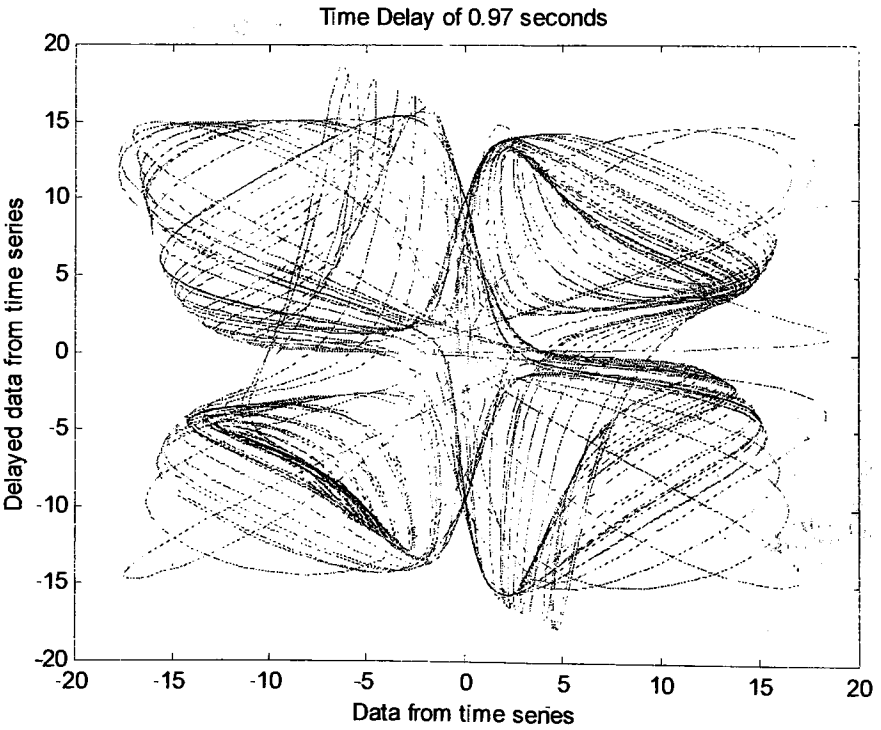


Figure 3.14: Reconstructed Attractor using First Local Minimum Autocorrelation Method

A third method that has been employed when using the autocorrelation function to find a delay time value to be used for attractor reconstruction. This method consists of finding the location where the autocorrelation of the signal is a fraction of the maximum value[32]. One chooses a delay time value that corresponds to a point where the autocorrelation is equal to one-half, $1/e^{th}$, or one-tenth of the value corresponding to a delay time of zero. Figure 3.10 shows that the delay time value for these three methods is equal to 0.26 seconds, 0.33 seconds, and 2.35 seconds, respectively. Figure 3.15 shows the attractor reconstructed using the delay time value where the autocorrelation was one-half of its maximum value. The reconstructed attractor using the delay time from the $1/e^{th}$ of the maximum value method is shown in Fig. 3.16. Figure 3.17 shows the attractor reconstructed using the delay time value obtained when the autocorrelation is equal to one-tenth of the maximum value. While these methods often work well when applied to a specific chaotic system, they usually do not provide a general approach.

The fourth method for determining a delay time value for attractor reconstruction using the autocorrelation function is the First Inflection Point Method[15,24,56]. This method locates the delay time value at the first inflection point of the autocorrelation function. This location corresponds to the first zero of the second derivative of the autocorrelation function. Figure 3.10 shows that for this Lorenz system this method produces the lowest delay value, 0.16 seconds. When used to reconstruct the system attractor, this delay value produces the best results of the four methods. The reconstructed attractor produced from this value can be seen in Fig. 3.18.

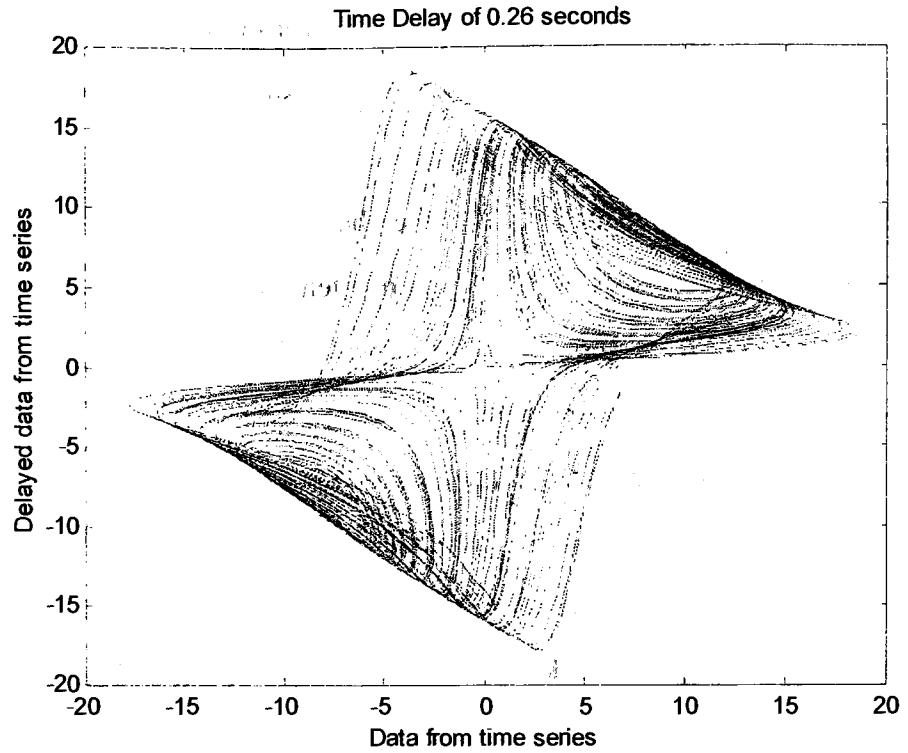


Figure 3.15: Reconstructed Attractor using Half Maximum Autocorrelation Method

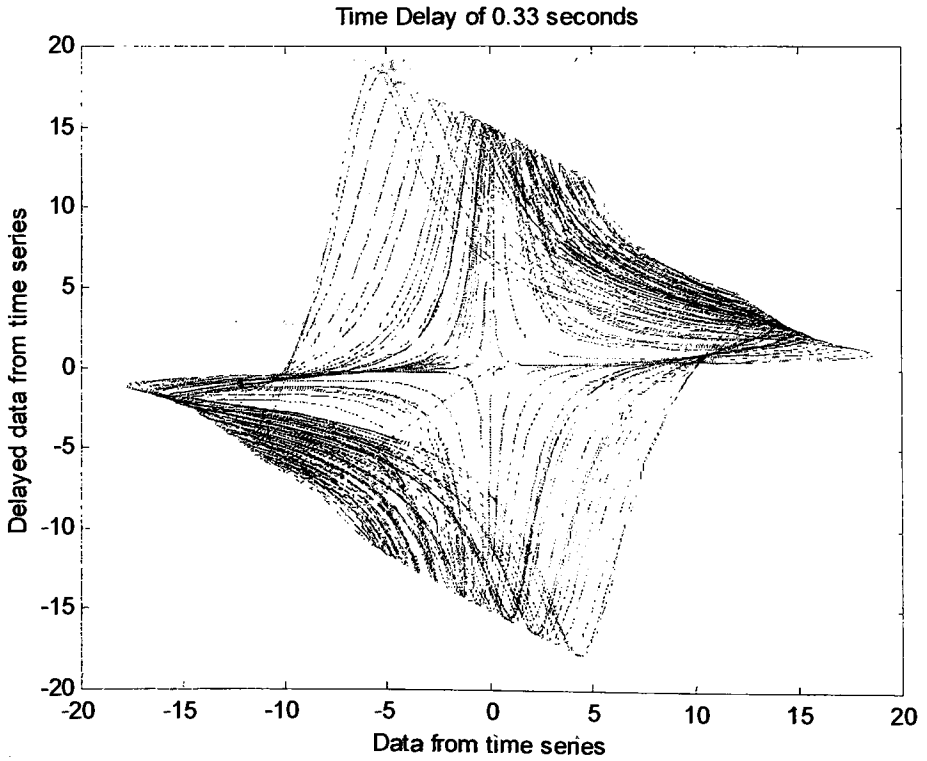


Figure 3.16: Reconstructed Attractor using $1/e^{th}$ of Maximum Value Method

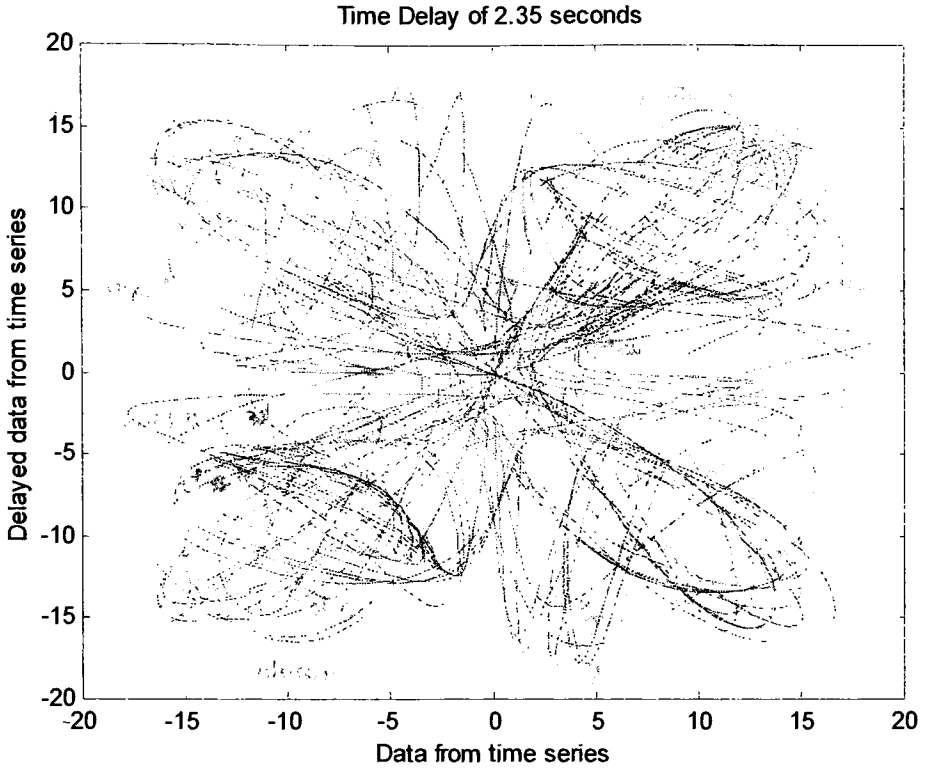


Figure 3.17: Reconstructed Attractor using One-tenth of Maximum Value Method

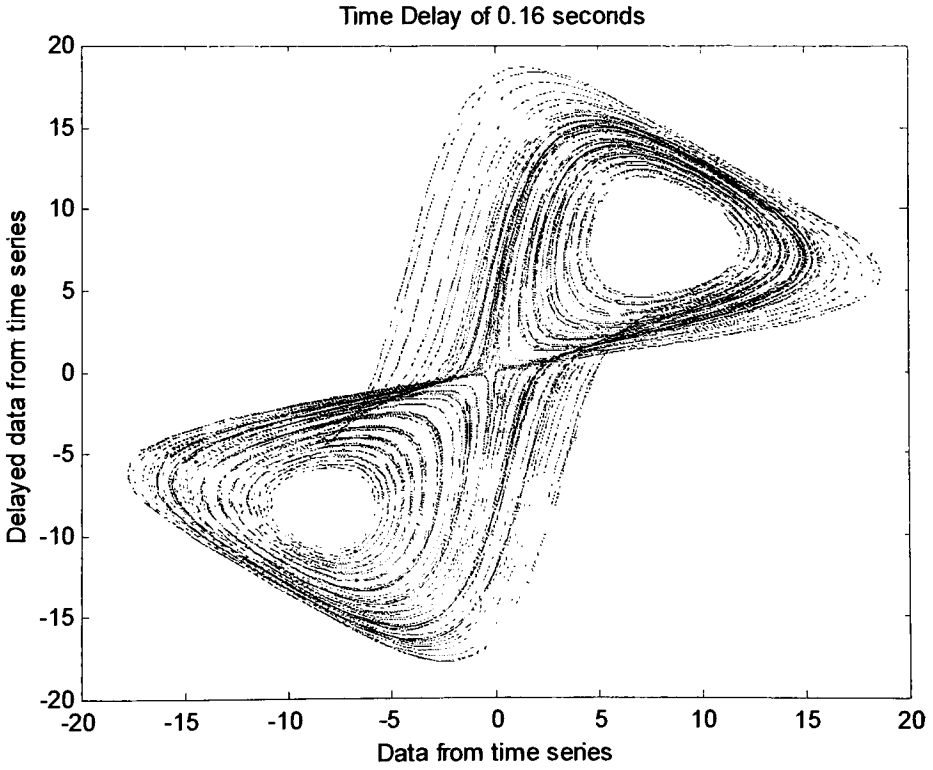


Figure 3.18: Reconstructed Attractor using First Inflection Point Autocorrelation Method

3.3.3 MUTUAL INFORMATION METHOD

Although the methods that use the autocorrelation function can determine a delay time quickly with few calculations, it only examines the linear relationship between the two data sets. Because these data sets contain data from nonlinear systems, this function is not thought to provide an adequate analysis of the data sets. Another method used to determine the delay time value for attractor reconstruction is the Average Mutual Information function[4,7,15,24,37,54]. This method works in a similar fashion as those using the autocorrelation function. It is thought to provide a better choice of the delay time because it examines a general relationship between the two data sets as opposed to only the linear relationship. This method is used to locate a delay time value for which the two sets of data are minimally dependent. The formula used to calculate the Average Mutual Information as a function of delay time is presented in Eq. 3.7.

$$I(\tau) = H(X) - H(X|Y) \tag{3.7}$$

To determine the Average Mutual Information using Eq. 3.7, two values must first be determined. These values, $H(X)$ and $H(X|Y)$ are the uncertainties in data sets. $H(X)$ is the uncertainty of x in isolation and $H(X|Y)$ is the uncertainty of x given a measurement of y. The last uncertainty must be calculated in two parts as shown in Eq. 3.8.

$$I(\tau) = H(X) + H(Y) - H(X,Y) \tag{3.8}$$

Since these uncertainties are based on the entropies within the data sets, the calculation of these uncertainties are done using the equation for entropy. When only one data set is being examined, the entropy is based on the one-dimensional set. When both data sets are being examined, the entropy is based on a two-dimensional space created from the two data sets. The equations for calculating the uncertainties are shown as Eq. 3.9 and Eq. 3.10.

$$H(X) = -\sum_i P_1(x_i) log_2(P_1(x_i)) \tag{3.9}$$

$$H(X,Y) = -\sum_i \sum_j P_2(x_i, y_j) log_2\left[P_2(x_i, y_j)\right] \tag{3.10}$$

Calculation of $H(Y)$ can be conducted using Eq. 3.9 by substituting the Y data set for the X data set. When a *log* of base two is used, the results of the calculations are in units of bits. In the previous two equations, the terms $P_1(\bullet)$ and $P_2(\bullet,\bullet)$ are the one-dimensional and two-dimensional probability distributions of the data. The one-dimensional probability distribution of the data is calculated in a similar manner to a histogram. The range over which the data occurs is divided into equal sections and the number of times that the data set falls within each of the sections is recorded. With the number of times the data set falls into each section and the total number of data points, the probability that a data point for the set will fall into each section can be calculated. The two-dimensional probability distribution is calculated in very similar manner. Using the two data sets, the data is plotted into a two-dimensional space with each data set along a perpendicular axis. The area occupied by the data is then divided into two-dimensional sections. As before, the number of times the data set falls into each section is counted. With this distribution and the total number of points within the data sets, it is possible to calculate the probability that a data point will fall within each of the sections. Figure 3.19 is a diagram that shows how these probabilities are determined.

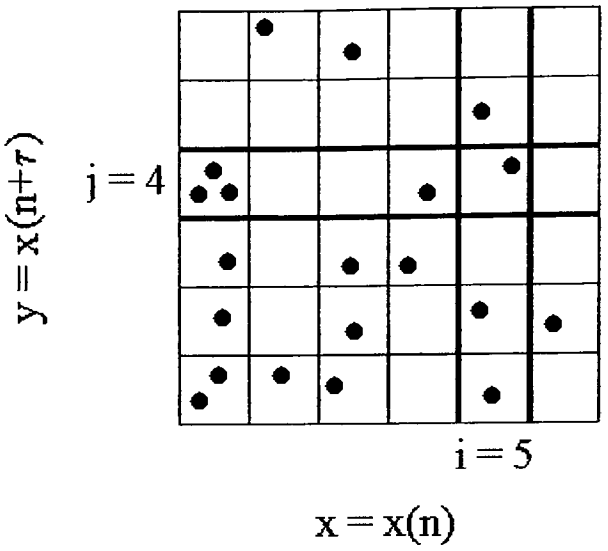


Figure 3.19: Average Mutual Information Calculation Diagram

The following are examples of probabilities as calculated from the data set of twenty points shown in Fig. 3.19. From the two-dimensional view of the two data sets, it is possible to calculate both the one-dimensional and two-dimensional probabilities. The one-dimensional probability is calculated by counting all the data points that fall into all the two-dimensional elements for one axis corresponding to one location on the other axis and dividing this value by the total number of data points. The two-dimensional probability is calculated by counting all the data points that fall into a single two-dimensional element divided by the total number of data points.

$$P_1(x_5) = \frac{4}{20} = 0.2 \tag{3.11}$$

$$P_1(y_4) = \frac{5}{20} = 0.25 \tag{3.12}$$

$$P_2(x_5, y_4) = \frac{1}{20} = 0.05 \tag{3.13}$$

This particular method for determining a delay time value for attractor reconstruction has been rigorously studied. Because the exact values produced by the Average Mutual Information function are of no importance and only the trends in the

data are used, it has been possible to modify and simplify the equation. Even if the results are multiplied by a constant, the trends within the data would remain. It is because of this, that the calculation of the Average Mutual Information can be simplified from four summations to two. The form of the Average Mutual Information function that is currently being used is shown in Eq. 3.14.

$$I(\tau) = \sum_{i=1}^{R}\sum_{j=1}^{S} P_2(x_i, y_j) log_2 \frac{P_2(x_i, y_j)}{P_1(x_i)P_1(y_j)} \tag{3.14}$$

$$\text{where } x_n = x(n) \text{ and } y_n = x(n-\tau)$$

In Eq. 3.14, R is the number of divisions for data set x and S is the number of divisions for data set y. The resulting curve created by this function is analogous to the output of the autocorrelation function. When the Average Mutual Information function produces a value of zero, the two data sets being examined are statistically independent. The data sets are statistically dependent when the value is positive. To determine the delay time to be used to reconstruct the system's attractor, the first local minimum is located. The curve produced using Eq. 3.14 over a range of delay time values can be seen in Fig. 3.20.

Once this curve has been produced, a simple forward differencing method can be used to locate the first local minimum of the curve. At this location, there is a relative minimum in the general relationship between the two data sets and for that reason, the corresponding delay time value will be used. The first local minimum for this data set from the Lorenz equations occurs at a delay time value of 0.27 seconds. The attractor reconstructed from this data set and delay time value is shown in Fig. 3.21.

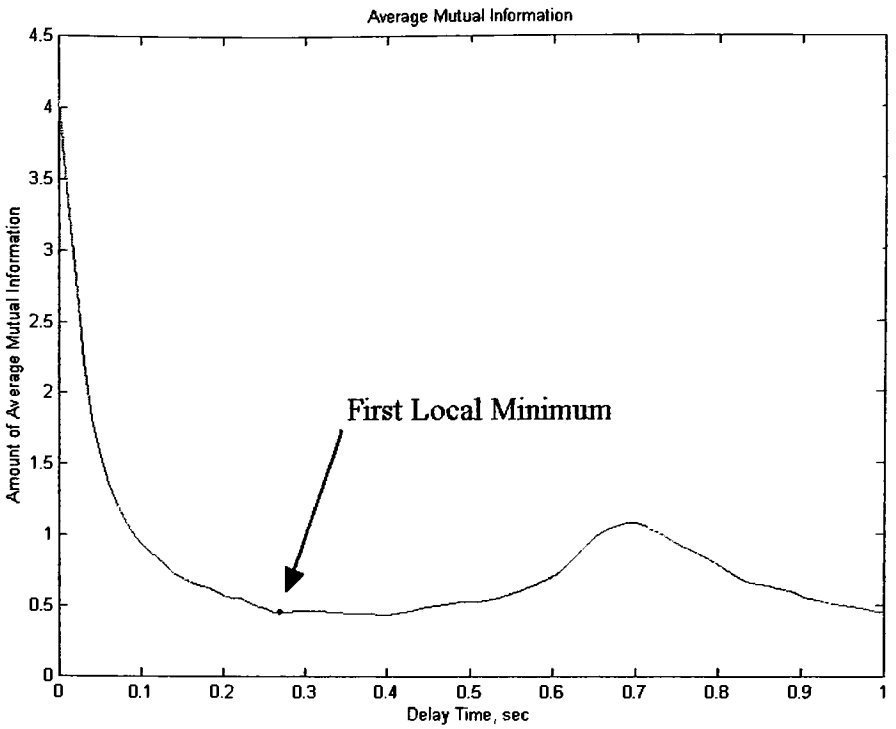


Figure 3.20: Average Mutual Information Curve

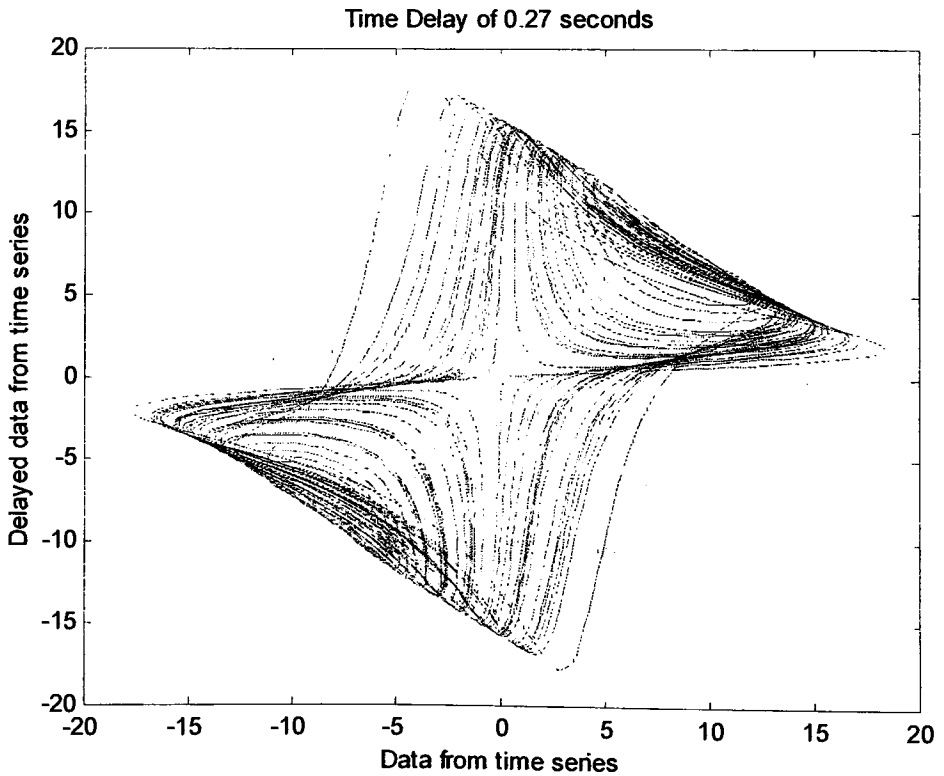


Figure 3.21: Reconstructed Attractor using Mutual Information Method

While this method is more computationally intensive than the methods employing the autocorrelation function, it is thought to produce better results. Most analysis currently being done on chaotic signals involving attractor reconstruction are equally distributed between methods that use the autocorrelation function and those based on the Average Mutual Information function.

3.3.4 AVERAGE DISPLACEMENT METHOD

Another method for determining a delay time value for attractor reconstruction is the Average Displacement Method. This method searches for a time delay by measuring the expansion of the reconstructed attractor across a range of delay times. The delay time is chosen so that the attractor is sufficiently expanded from the bisector line. The Average Displacement is determined by summing the difference between the first component of the vector and all the other components and dividing the value by the total number of reconstructed vectors. This calculation is done with the formula given in Eq. 3.15.

$$S(\tau) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\sum_{j=1}^{d_e-1}(x_{i+j\tau} - x_i)^2} \qquad (3.15)$$

In this formula, τ is the delay time, d_e is the embedding dimension, N is the length of the reconstructed data set, and x_i are values from the original data set. As the delay value is increased from zero, the reconstructed trajectory expands from the 45° line where $x(n) = x(n+\tau)$. The value of the Average Displacement, corresponding to the distribution of the data set throughout the reconstructed space, will increase until it reaches a level where the data set is as widely distributed as possible. When this level is reached, the value of the Average Displacement function becomes saturated. A data curve produced by this method is shown in Fig. 3.22. The data set used for this

calculation is the same x data set collected from the Lorenz equations used in the previous methods.

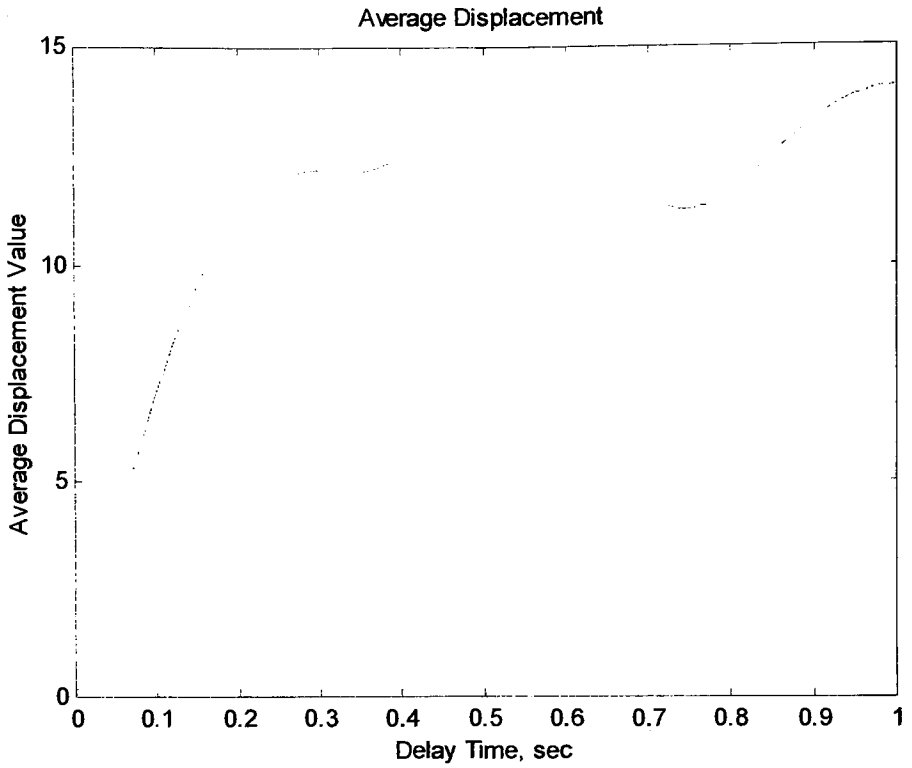


Figure 3.22: Average Displacement Curve

Use of this method has shown that the optimal delay time occurs where the slope of the average displacement curve decreases to less than forty percent of its initial value [34]. To determine this location, it is necessary to calculate the first derivative of the curve produced by the Average Displacement function. The derivative is calculated using the simple discrete derivative formula in Eq. 3.16.

$$\frac{dS}{d\tau}(\tau) = \frac{S(\tau+1) - S(\tau-1)}{2*\Delta\tau} \tag{3.16}$$

By applying this formula to the curve shown in Fig. 3.22 produced by the Average Displacement function, it is possible to produce a curve displaying the slopes at each point of the original curve. With this new curve, the initial slope of the average displacement curve is known and it is possible to determine when the slope is less than

forty percent of this value. Figure 3.23 shows the derivative of the curve produced by the Average Displacement function.

Using the Average Displacement method, Fig. 3.23 shows that the slope of the average displacement curve falls below forty percent of the initial slope at a delay value of 0.18 seconds. The value is then used in the reconstruction of the attractor using the data set that was examined. The reconstruction produces the image shown in Fig 3.24.

This method appears to produce an adequate delay value for reconstructing a strange attractor, but the reconstruction still contains distortions where the orbits of the trajectory are highly concentrated. The method also supports another theory regarding attractor reconstruction: when the value of the embedding dimension is increased, a smaller delay time value is produced. It has been proposed that the delay time value is not independent of the embedding dimension[38]. It is suggested that the delay window, τ_w, should be determined. The value of the delay time would then be a function of this value and the prescribed embedding dimension. The **delay window** is defined by Eq. 3.17.

$$\tau_w = (d_e - 1)\tau \tag{3.17}$$

This value corresponds to the total amount of delay between the first component and the last component in the reconstructed vector as shown in Eq. 3.2. When the value of the delay window is kept constant, the delay value produced will diminish as the embedding dimension increases.

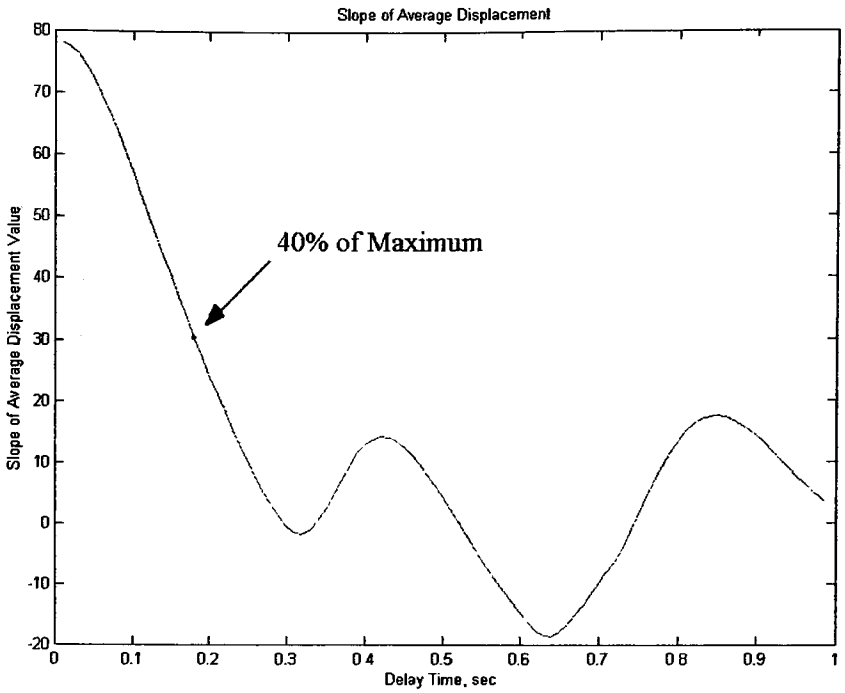


Figure 3.23: First Derivative of Average Displacement Curve

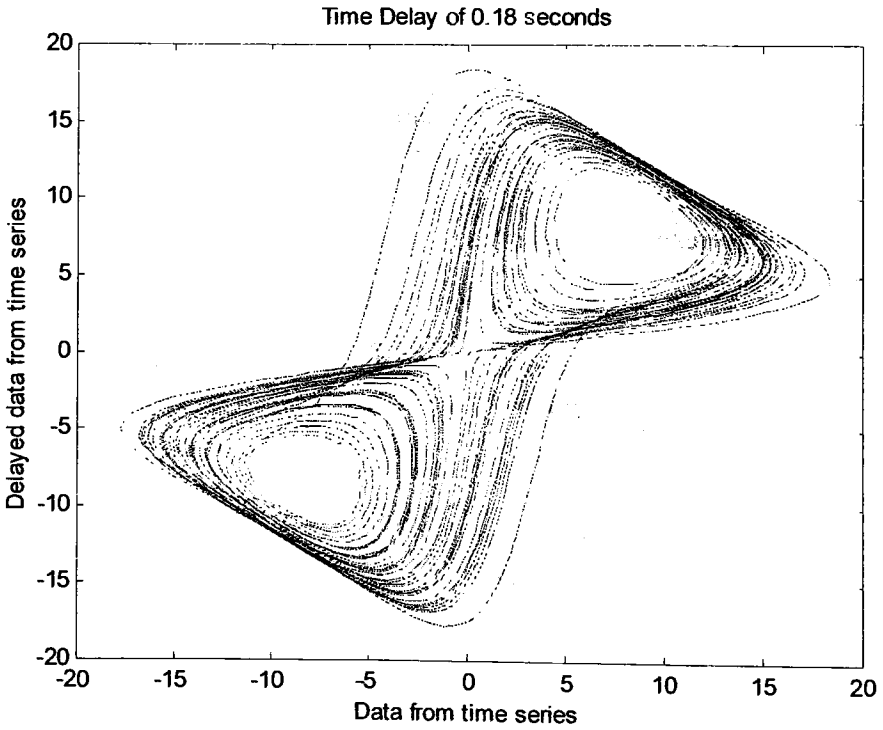


Figure 3.24: Attractor Reconstructed using Average Displacement Method

3.4 SINGULAR SYSTEM APPROACH

The Singular System Approach is a method of attractor reconstruction that does not require determining a delay time value and is also effective in removing additive noise from data[18]. This method works very well on actual data collected from chaotic systems. The data from the available time series is arranged into a matrix with a delay time equal to the sampling rate and embedded with a dimension of n. This embedding produces a **trajectory matrix** using the formula in Eq. 3.18.

$$[\boldsymbol{X}] = N^{-\frac{1}{2}} \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \vdots \\ \underline{x}_N^T \end{bmatrix} = N^{-\frac{1}{2}} \begin{bmatrix} x(1) & x(2) & \dots & x(n) \\ x(2) & x(3) & \dots & x(n+1) \\ \vdots & \vdots & & \vdots \\ x(N) & x(N+1) & \dots & x(N+n-1) \end{bmatrix} \tag{3.18}$$

The value of n used to reconstruct the strange attractor is determined from the previously mentioned delay window. After the value for the delay window is calculated, it is divided by the sampling time to produce a value of n for the creation of the trajectory matrix. King, Jones, and Broomhead state that the delay time of the first inflection point of the autocorrelation of the data provides a satisfactory delay window[18]. The value of N in the formula is equal to $N_T - n + 1$, where N_T is the length of the original data set. King, Jones, and Broomhead also stated that Takens' embedding inequality should be altered to state that $d \geq 2m + 1$ and $d < n$, where m is the dimension of the underlying manifold and d is the rank of the trajectory matrix. In the absence of noise, d would be equal to the number of linearly independent rows or columns. When noise is present, d is equal to the number of singular values that are above the "noise floor". The **noise floor** is the level where the values of the singular values obtained from the singular value decomposition of the trajectory matrix level off. When there is noise combined with the signal, the values of all the singular values are uniformly increased. The values that would be equal to zero are instead raised up to what is known as the noise floor. This concept is shown in Fig. 3.25.

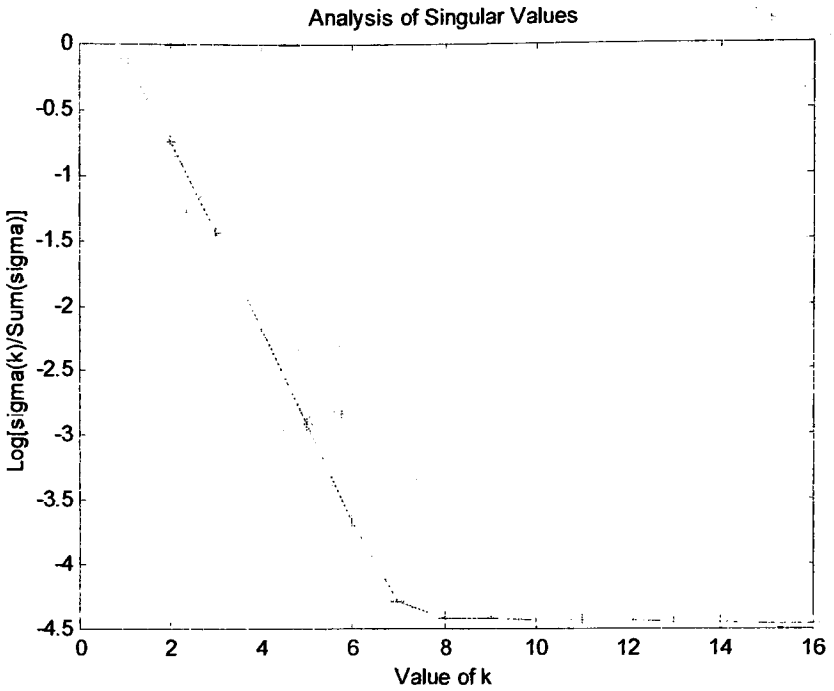


Figure 3.25: Logarithmic Analysis of Singular Values

After the matrix has been constructed, the data is decomposed using the concept of Singular Value Decomposition. Singular Value Decomposition allows the matrix $[\boldsymbol{X}]$ to be factored into three new matrices. This relation is shown in Eq. 3.19.

$$[\boldsymbol{X}] = [S][\Sigma][C]^T \tag{3.19}$$

Here $[S]$ and $[C]$ are square matrices with orthogonal columns. The columns of matrix $[C]$ are the eigenvectors of the transpose of the trajectory matrix multiplied by itself. Matrix $[\Sigma]$ is a diagonal matrix in which the diagonal elements are the singular values from the original matrix. The jth singular value is represented by σ_j, and can be calculated using Eq. 3.20.

$$\sigma_j = \left[N^{-1}\sum_{i=1}^{N}\left(\underline{x}_i^T \cdot \underline{c}_j\right)^2\right]^{\frac{1}{2}} \tag{3.20}$$

In this equation, $\underline{c}_j$ is the jth column of the orthogonal matrix $[C]$. After the Singular Value Decomposition of the trajectory matrix is complete, the singular vectors and singular values of the trajectory matrix are known. The vectors are used to establish

a new basis to accommodate the most populated directions using a least squares approach. The values produced are related to the degree to which the data will fill the new directions. Analysis of these singular values reveals the rank of the matrix. First, the sum of all the singular values is calculated. Then each singular value is divided by this sum to determine what percent of the total each singular value represents. Once this is done, the natural logarithm of each of these percentages are calculated and plotted against the indices of the singular values. Figure 3.25 shows an example of this plot.

Figure 3.25 indicates that there are two distinct sets of singular values. Those that exhibit a rapid decrease in value are the deterministically dominated set. The size of the deterministically dominated set is equal to the **rank**, d, of the matrix. This particular data set has a rank of $d = 7$. The singular values on the right-hand side of the figure are the noise dominated set. The noise dominated set is distinguished by the nearly constant value of the singular values. After the two sets of singular values have been identified, examination of the singular vectors can be used to verify the results. Figure 3.26 and Fig. 3.27 show the singular vector components corresponding to the first two singular values. Each vector produces a smooth curve without any abnormalities. Figure 3.28 shows the components of the singular vector corresponding to the last singular value in the set of deterministic dominated set. Comparison of this vector with the singular vector in Fig. 3.29 for the first singular value in the noise dominated set shows that the noise dominated singular vectors are not smooth like the vectors that display deterministic characteristics. Figure 3.30 and Fig. 3.31 show additional singular vectors corresponding to the singular values of the noise dominated set.

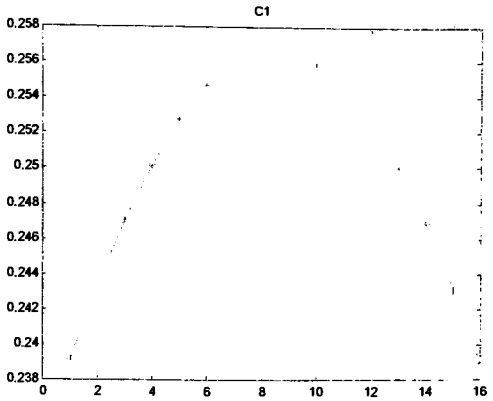


Figure 3.26: C_1

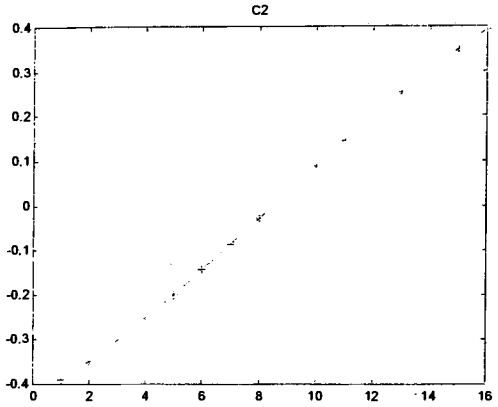


Figure 3.27: C_2

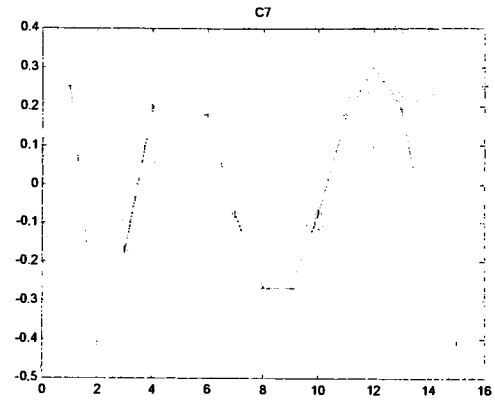


Figure 3.28: C_7

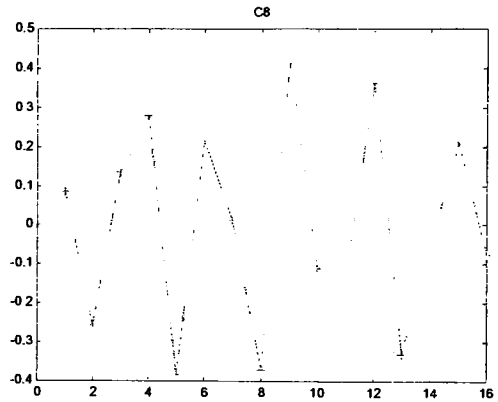


Figure 3.29: C_8

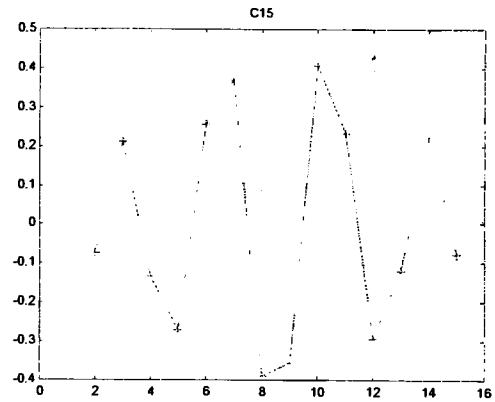


Figure 3.30: C_{15}

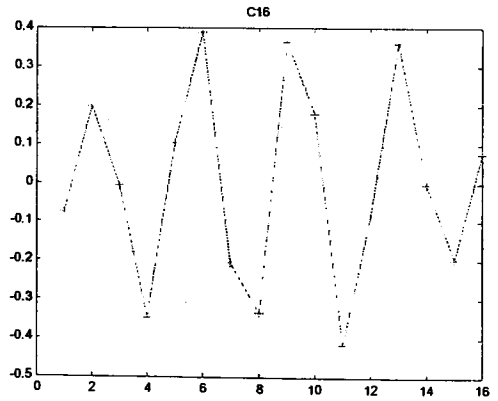


Figure 3.31: C_{16}

When the vectors of matrix $[\boldsymbol{X}]$ are projected onto the d most populated directions, it produces a d-dimensional reconstruction of the system's strange attractor. The $(n - d)$ vectors that are discarded include the abnormal variations and noise present

within the system. The method for projecting the trajectory matrix onto the state space created using the orthogonal vectors from matrix $[C]$ is discussed in the following paragraphs.

Each row of the trajectory matrix is represented as the vector $\underline{x}_i^{\mathrm{T}}$, where each vector has n components. When the vector from the trajectory matrix is plotted into the state space, a new vector is created. The components of this new column vector, $\underline{z}_i$, correspond to the amount of each of the orthogonal column vectors of $[C]$ that exist within the original vector from the trajectory matrix. This relation is shown as Eq. 3.21. To solve for the vector $\underline{z}_i$, both sides of the equation must be multiplied by the inverse of matrix $[C]$.

$$[C]\{\underline{z}_i\} = \{\underline{x}_i\} \tag{3.21}$$

$$\{\underline{z}_i\} = [C]^{-1}\{\underline{x}_i\} \tag{3.22}$$

Since matrix $[C]$ is orthogonal, the transpose of this orthogonal matrix is equal to the inverse of the matrix. Taking the transpose of both sides of the Eq. 3.22 produces the equation for the row vector $\underline{z}_i^T$.

$$\{\underline{z}_i\} = [C]^T\{\underline{x}_i\} \tag{3.23}$$

$$\{\underline{z}_i\}^T = \left[[C]^T\{\underline{x}_i\}\right]^T \tag{3.24}$$

$$\{\underline{z}_i\}^T = \{\underline{x}_i\}^T[C] \tag{3.25}$$

Using the rule governing the transpose of a product of two matrices, Eq. 3.25 shows that the new row matrix is merely the product of the original vector and the matrix of orthogonal column vectors. From this relationship, the time series of the new vectors is determined through matrix multiplication of the trajectory matrix and matrix $[C]$, the orthogonal column vectors.

$$[\boldsymbol{Z}] = \begin{bmatrix} \underline{z}_1^T \\ \underline{z}_2^T \\ \vdots \\ \underline{z}_N^T \end{bmatrix} = [\boldsymbol{X}][\boldsymbol{C}] \tag{3.26}$$

After this process has been complete, the first d columns of the new matrix can be used to produce the reconstruction of the strange attractor corresponding to the system from which the single variable time series was collected. Figure 3.32 shows the two-dimensional projection of the reconstructed attractor produced using the Singular System Approach.

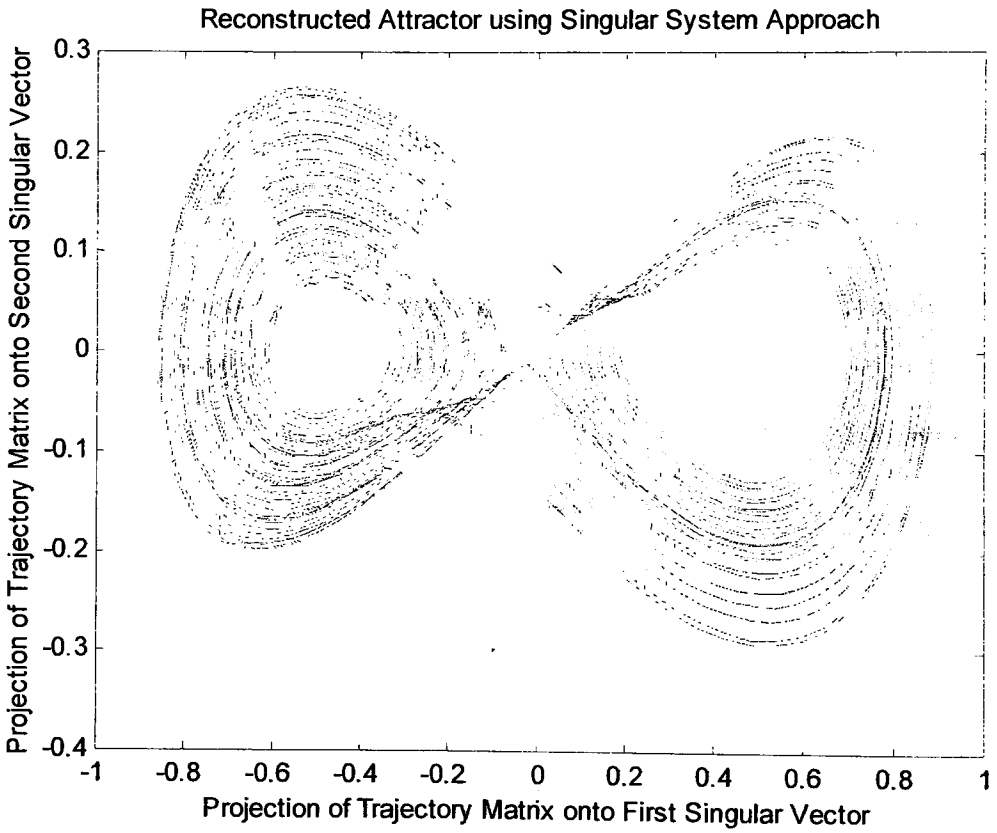


Figure 3.32: Reconstructed Attractor Using Singular System Approach

When an optimal value of n is selected during the creation of the trajectory matrix, the results of the method appear to produce a successful reconstruction of the original strange attractor. The choice of n must be large enough so that an adequate

number of vectors exist to successfully separate the deterministic portion of the signal from the noise dominated portion of the signal. However if the value of n is too large, the deterministic components of the signal are spread out over more vectors than necessary. This results in a shorter time series of the reconstructed vectors and greatly increases the amount of time required for subsequent analysis. Figure 3.33 shows a reconstruction of the attractor when the embedding of the trajectory matrix is very small. While the attractor does contain geometric similarities to the true attractor, it also contains a significant amount of distortion. The attractor shown as Fig. 3.34 was reconstructed using an n value much larger than necessary. This two-dimensional projection does not contain as much information about the full attractor as possible with a smaller embedding dimension. This can be seen where the paths of the trajectory have become very concentrated. The reconstructed attractor must be examined in a much higher dimensional space to gain all the deterministic information of the signal.

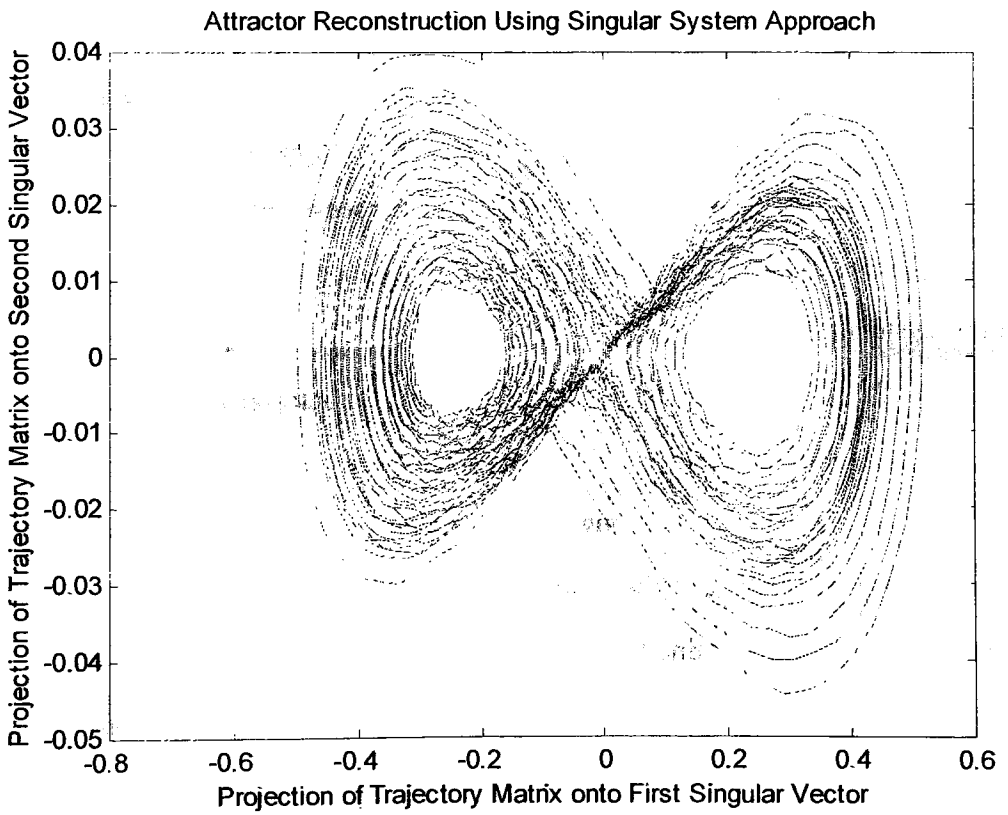


Figure 3.33: Too Low of a Value of n

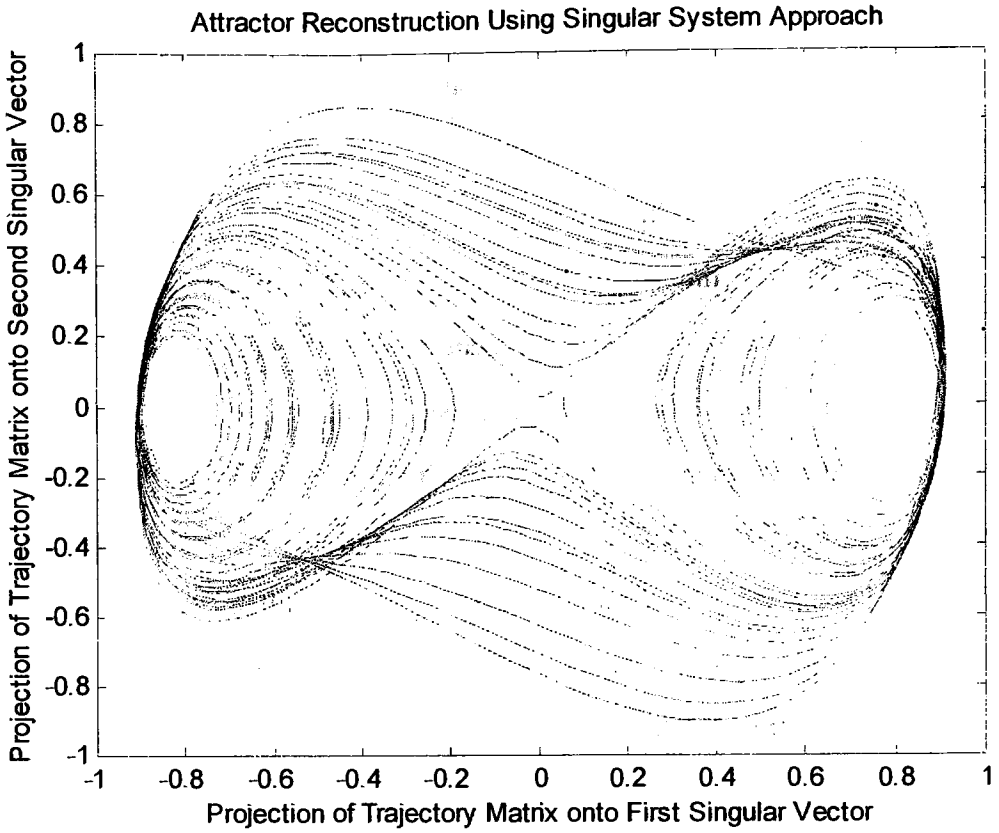


Figure 3.34: Too High of a Value of n

Despite how effective this method appears to be, similar to the autocorrelation function, Singular Value Decomposition is a linear process, and as a result, the Singular System Approach it is thought by some as a misleading technique when used on nonlinear systems[22].

With the ability to reconstruct the system's attractor from the time series of a single state variable, it is possible to perform various analyses on actual physical systems exhibiting chaotic behavior. Although it is possible to determine an optimal delay time value to reconstruct a system's attractor using trial and error, it is important to have a systematic method to determine this value so that it can be implemented using computer software. Once it is possible to consistently determine an optimal means for attractor reconstruction, analysis of a system from a single state variable time series will become more efficient and more reliable results can be produced.

4 FRACTAL DIMENSION

4.1 GENERAL

Once an attractor has been reconstructed, one property that can be estimated is the dimension of the attractor. The **dimension** is the minimum number of coordinates required to describe every point within the data set[41]. In the case of a point, the dimension is zero. A curve has a dimension of one and an area has a dimension of two. In the same manner, a volume will have a dimension of three. Along with structures that have integer dimensions, there also exists data sets with non-integer dimensions. These non-integer dimension data sets are referred to as having a **fractal dimension**. This fractal dimension is a result of the fractal structure of the data set. A **fractal** is a complex geometric shape with fine structure at arbitrarily small scales. Chaotic attractors have fractal dimensions.

There are many different ways to calculate the dimension of a chaotic attractor. Some of these methods include the Similarity Dimension, Capacity Dimension, Information Dimension, Correlation Dimension, and Lyapunov Dimension. Some of these methods are more sensitive to the distribution of the data points throughout the state space and some are able to calculate the dimension more efficiently. Calculation of the dimension of a data set can be used to distinguish chaos from stochastic noise, distinguish between different chaotic attractors, and quantify the level of chaos within the data set. The further the value of the fractal dimension is away from an integer, the more chaotic the data set.

4.2 SIMILARITY DIMENSION

The Similarity Dimension is a specific type of dimension that applies to self-similary fractal structures[14]. **Self-similarity** is a characteristic of a fractal where details at one scale are similar, but not necessarily identical, to structures seen on

different scales[3]. This unique property enables the dimension of the data set to easily be determined. The two values required to calculate this dimension are m, the number of copies of the original fractal and r, the scale factor. Using these values, the Similarity Dimension, D_{sim}, is calculated using the following formula.

$$D_{sim} = \frac{ln(m)}{ln(r)} \tag{4.1}$$

The Similarity Dimension is based on the relationship between the number of copies and the scale factor of the copies. The exponential relation between these two values is shown as Eq. 4.2.

$$m = r^{D_{sim}} \tag{4.2}$$

Examination of a structure with an integer dimension can be used to support this method. Calculation of the dimension is performed in the same manner with integer dimensions as it is when examining self-similar fractal structures. When a line is examined, the values of both m and r are equal to two. This produces a Similarity Dimension of one. The values of m and r for a square area are equal to four and two, respectively. This produces a dimension of two. Similarly, when a cube is examined, an m value of eight and an r value of two produce a dimension of three. The dimension of a square is calculated using the subdivision in Fig. 4.1.

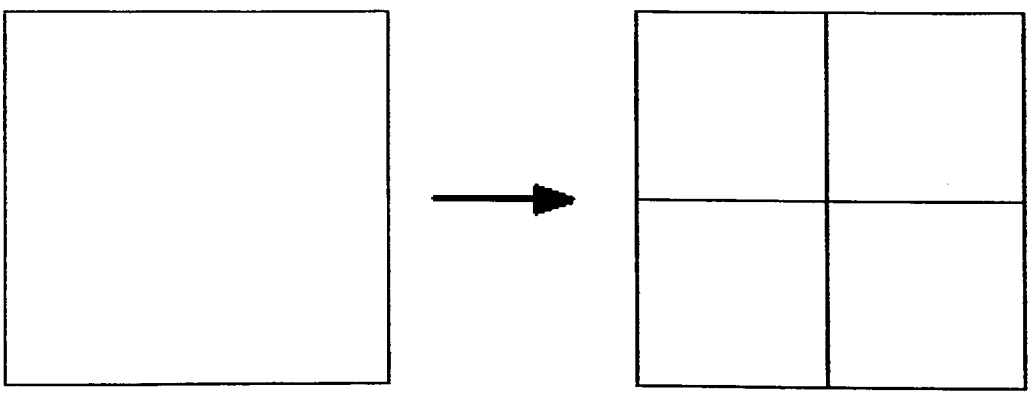


Figure 4.1: Similarity Dimension of an Area

The area of a square must be divided into a number of equal sized pieces that maintain the same geometric shape as the original structure. This square is divided into four smaller squares. As a result, there are four copies of the original structure that have lengths equal to half the original size. With m equal to four and r equal to two, the Similarity Dimension is equal to two, supporting this method. The following equation shows the calculation of this dimension using this example.

$$D_{sim} = \frac{ln(4)}{ln(2)} = \frac{1.386}{0.693} = 2 \tag{4.3}$$

The fractal dimension of self-similar fractals is determined the same way. The only difference is that the dimension is a non-integer number. One particular fractal structure for which the Similarity Dimension can be readily calculated is the *von Koch curve*. The ***von Koch curve*** is a curve created by removing the middle third of a line segment and replacing it with two segments at equal angles with respect to the original line segment[14]. This process is then repeated for the four new line segments and again with each of the sixteen line segments. After the process has been repeated an infinite number of times, the *von Koch curve* will have an infinite arc length. Because of this characteristic, the dimension of this curve is not equal to one. Figure 4.2 shows how the *von Koch curve* is created. This information is used to determine the dimension of the curve using the Similarity Dimension method.

Figure 4.2: Similarity Dimension of von Koch curve

Examination of the *von Koch curve* shows that after each step, the curve has been altered such that it consists of four copies of the original curve at a scale of one-third.

This results in a value of four for m and a value of three for r. This produces a Similarity Dimension of about 1.26.

$$D_{sim} = \frac{ln(4)}{ln(3)} = \frac{1.386}{1.099} = 1.262 \tag{4.4}$$

4.3 CAPACITY DIMENSION

Another type of dimension that is very much like the Similarity Dimension is the Capacity Dimension. The Capacity Dimension, also known as the Box Counting Dimension or Hausdorff Dimension, examines the number of n-dimensional elements required to cover the entire n-dimensional data set[14,31]. As the size of the n-dimensional elements are decreased, the number of elements required to cover the data set will increase at an exponential rate. This relation is shown as Eq. 4.5.

$$N(\epsilon) = k\epsilon^{-D_{cap}} \tag{4.5}$$

In this equation, D_{cap} is the Capacity Dimension, ϵ is the diameter of the volume elements, $N(\epsilon)$ is the number of ϵ sized volume elements, and k is a constant dependent on the geometry of the data set and the type of volume element used. To calculate the Capacity Dimension, Eq. 4.5 is rearranged to solve for the dimension. Equation 4.6 displays the definition of the Capacity Dimension of a given data set.

$$D_{cap} = \lim_{N \to \infty} \frac{ln\left(\frac{N(\epsilon)}{k}\right)}{ln\left(\frac{1}{\epsilon}\right)} \tag{4.6}$$

To determine the Capacity Dimension, the number of volume elements must be determined for a range of different element sizes. Once this is done, the number of elements can be plotted against the element sizes on a log versus log graph. The slope of the linear portion of this curve is equal to the Capacity Dimension of the data set. The extremes of the curve are nonlinear due to the structure of the data signal being examined. When the element size is very small, the size of the elements will be smaller

than the distance between any two data points and each element will only contain one data point. When the element size approaches the diameter of the attractor, a few elements will enclose all of the data points. Eventually, one single volume element will enclose the data set. Because of this behavior at each extreme of the curve, the relationship in Eq. 4.5 will not hold. Figure 4.3 shows how the dimension is obtained from the linear portion of this graph.

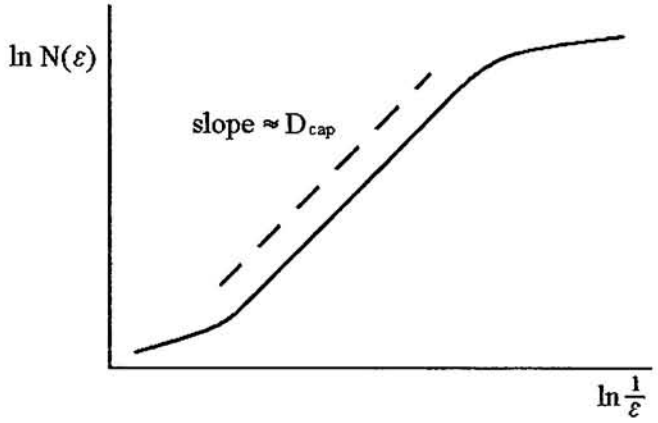


Figure 4.3: Determination of Capacity Dimension

A structure with a known integer dimension can be used to support the Capacity Dimension method. Figure 4.4 shows how the Capacity Dimension method can be used to determine the dimension of a two-dimensional data set.

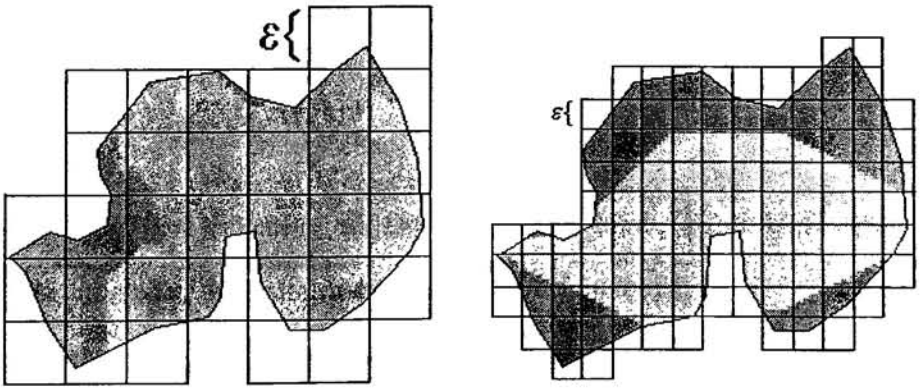


Figure 4.4: Capacity Dimension of Area

After determining the number of two-dimensional elements required for a range of element sizes, the relationship between the number of elements and the size is

determined. Equation 4.7 displays this relationship. The relationship in Eq. 4.7 can then be substituted into Eq. 4.6 to verify the dimension of the area.

$$N(\epsilon) \propto \frac{A}{\epsilon^2} \tag{4.7}$$

$$D_{cap} = \lim_{\epsilon \to 0} \frac{ln(N(\epsilon))}{ln\left(\frac{1}{\epsilon}\right)} \approx \lim_{\epsilon \to 0} \frac{ln\left(\frac{A}{\epsilon^2}\right)}{ln\left(\frac{1}{\epsilon}\right)} = 2 \tag{4.8}$$

4.4 INFORMATION DIMENSION

While the Capacity Dimension examines the area that is inhabited by the data set, it does not take into account the density of the data points. The **seniority**, or concentration of data points within each element can be used to further differentiate fractals that occupy the same volume within state space[11]. The Information Dimension and the Correlation Dimension take into account the distribution of the data points as well as the volume of state space that they occupy.

The Information Dimension examines the change in entropy of a set of data points as the size of the volume elements changes[31]. **Entropy** is the amount of information required to specify the state of the system to an accuracy of ϵ if the state is known to be on the attractor. The entropy, $H(\epsilon)$, of the data set is calculated using Eq. 4.9.

$$H(\epsilon) = -\sum_{i=1}^{N(\epsilon)} P_i \, ln(P_i) \tag{4.9}$$

In this equation, P_i is the probability that the trajectory will pass through the ith volume element. This is calculated by dividing the number of points in the ith volume element by the total number of data points. The relation between the entropy of the data set and the size of the elements is similar to the relationships that define the Similarity Dimension and the Capacity Dimension. Equation 4.10 shows this relationship.

$$e^{H(\epsilon)} \approx k\epsilon^{-D_I} \tag{4.10}$$

This relationship shows that the amount of information required to specify the state increases inversely with the D_Ith power of ϵ[31]. The same rearrangement method as before is then used to define the Information Dimension. The definition of the Information Dimension is given in Eq. 4.11.

$$D_I = \lim_{\epsilon \to 0} \frac{H(\epsilon)}{ln(1/\epsilon)} \tag{4.11}$$

The Information Dimension can be determined by calculating the entropy using a range of different volume element sizes. Once they have been calculated, the entropy and element size are plotted onto a semi-log graph. After the values have been plotted, the slope of the linear section of this curve will be equal to the Information Dimension. This is done in the same fashion as it was illustrated for the Capacity Dimension in Fig. 4.3.

4.5 CORRELATION DIMENSION

Despite the effectiveness of the previously discussed methods of determining the fractal dimension of a data set, mapping out the volume elements can be computationally expensive. Another method for determining the fractal dimension that does not require mapping out volume elements is the Correlation Dimension[11,14,24,31,56]. The Correlation Dimension focuses on the concentration of the data points and how close they are to one another. As with the previous methods for determining the dimension, there is an exponential relationship between the radius around the point being examined, r, and the dimension. This relationship can be seen in Eq. 4.12.

$$C(r) \propto r^{D_{cor}} \tag{4.12}$$

In the relationship, $C(r)$ is the correlation sum of the data set. This value is dependent on how many points are within a given radius of each point. The definition of the correlation sum is given in Eq. 4.13.

$$C(r) = \left(\frac{1}{(N-1)N}\right) \sum_{i,j=1 i\neq j}^{N} H(r - ||x_i - x_j||) \tag{4.13}$$

Within the correlation sum, $H(\bullet)$ is the Heaviside function. The N within the equation is the number of data points in the data set, r is the radius around each point that is being examined, and $||x_i - x_j||$ is the Euclidean distance between two data points. After the correlation sum is calculated, the Correlation Dimension of the data can then be determined. The definition of the Correlation Dimension is given in Eq. 4.14.

$$D_{cor} = \lim_{\epsilon \to 0} \frac{ln\, C(r)}{ln\, r} \tag{4.14}$$

From this definition, the Correlation Dimension can be calculated for a set of data. The correlation sums and the corresponding range of radii being examined are plotted onto a log versus log graph. The slope of the linear portion of the curve is equal to the Correlation Dimension. This is also done in the same fashion as the previous methods for determining the dimension of a data set. The Correlation Dimension method is currently a preferred method because it can be applied to data sets of any dimension, it provides a more detailed examination than the Capacity Dimension, and it requires less calculations than the Information Dimension[27,35,49,54].

Through further work done with the Correlation Dimension by James Theiler, a modified version of the correlation sum was developed[43]. Theiler's addition to the correlation sum eliminates anomalous structures from the log graph of the correlation sums versus the corresponding radii. Equation 4.15 shows the modified correlation sum developed by Theiler.

$$C(r, N, W) = \frac{2}{N^2} \sum_{n=W}^{N} \sum_{i=1}^{N-n} H(r - ||x_{i+n} - x_i||) \tag{4.15}$$

Theiler found that while calculating the Correlation Dimension, the logarithmic graph of the correlation sums versus the radii was displaying an anomalous structure as a

result of a linear bias for small radius values. This problem was caused by the analysis of the attractor that included other nearby data points that were on the same trajectory orbit as the point being examined. Theiler proposed that by omitting this segment of the trajectory from the analysis, the linear bias would be removed and thereby the anomalous structure. Figure 4.5 shows how Theiler's modifications affected the data points on an attractor.

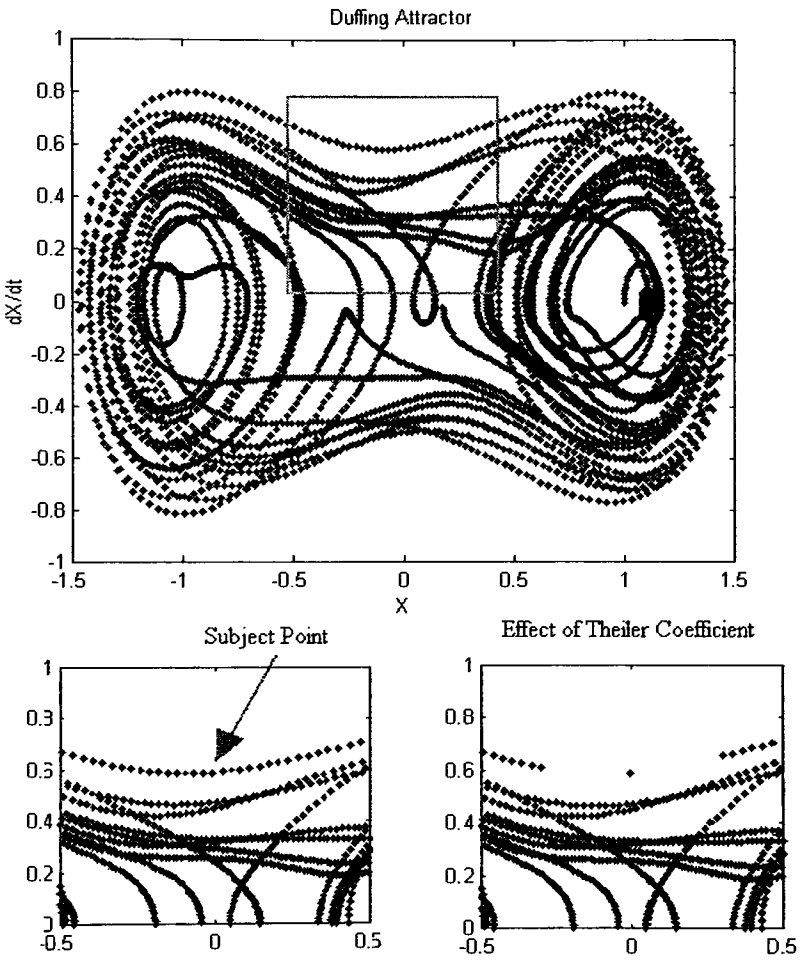


Figure 4.5: Effect of Theiler coefficient on a Strange Attractor

4.6 LYAPUNOV DIMENSION

The final method for determining the dimension of a data set that will be discussed is the Lyapunov Dimension[24,31]. This value requires the Lyapunov exponents of the associated dynamic system to first be determined. After the Lyapunov exponents have been calculated and arranged in order from the largest to smallest, the largest set of exponents must be determined so that their sum is positive. This relationship is displayed in Eq. 4.16.

$$\sum_{i=1}^{j}\lambda_i > 0 \text{ and } \sum_{i=1}^{j+1}\lambda_i < 0 \tag{4.16}$$

Equation 4.17 is the Lyapunov Dimension as defined by Kaplan and Yorke using the value of j that was determined from Eq. 4.16[12,14].

$$D_L = j + \frac{\sum_{i=1}^{j}\lambda_i}{|\lambda_{j+1}|} \tag{4.17}$$

Although each of the methods use different means to determine the fractal dimension of a set of data, they produce values that are very similar. Grassberger and Procaccia determined the relation between many of these values[12,31]. The relation between a number of the different methods is described by Eq. 4.18.

$$D_{LB} \leq D_{cor} \leq D_{cap} \leq D_L \tag{4.18}$$

This set of relations shows that the Lyapunov Dimension method produces the largest value. Following this value is the Capacity Dimension method and then the Correlation Dimension method. All of these are greater than or equal to the Lower Bound Dimension, D_{LB} which is equal to the number of non-negative Lyapunov exponents. Their study also indicated that the Information Dimension would produce a value between the Capacity Dimension and the Lower Bound Dimension.

$$D_{LB} \leq D_I \leq D_{cap} \tag{4.19}$$

5 DATA SETS AND COMPUTATION

5.1 GENERAL

To examine how effective each of the different methods is for attractor reconstruction, it was necessary to perform Lyapunov exponent and fractal dimension analyses on a number of different data sets. The data sets that were examined fall into two categories. The majority of the data sets were created by simulating various nonlinear systems that exhibited chaotic behavior for a range of different parameter values. The remaining data sets were collected from actual physical systems designed to produce chaotic behavior.

5.2 SIMULATED DATA

In order to validate each of the algorithms, three common nonlinear systems were selected. Each of these three systems was simulated using the most widely studied parameter values. By restricting the simulated data to these systems, the results of many of the algorithms could be compared to published values. The three systems that were utilized in this study were the Lorenz equations, the Rössler equations, and the Duffing equation.

The Lorenz equations were developed by Edward Lorenz in 1963 at the Massachusetts Institute of Technology to model atmospheric convection[23]. The Lorenz equations consist of three first-order differential equations that include two nonlinearities. These equations were presented in Chapter One and are shown again below.

$$\frac{dx}{dt} = \sigma(y - x) \tag{5.1}$$

$$\frac{dy}{dt} = rx - y - xz \tag{5.2}$$

$$\frac{dz}{dt} = xy - bz \tag{5.3}$$

The behavior of the Lorenz equations are governed by the three control parameters, σ, r, and b. When studied by Lorenz, the parameter σ was called the Prandtl number and r was referred to as the Rayleigh number. The third parameter was not given a name. This study examined the Lorenz equations for two common sets of parameter values. The first set will be considered the classical set and was the main focus of Edward Lorenz's research. The values for this set are $\sigma = 10$, $r = 28$, and $b = 8/3$. The other set of parameter values frequently used are $\sigma = 16$, $r = 45.92$, and $b = 4$. Various projections of the Lorenz attractor for the second set of parameter values are shown in Fig. 5.1.

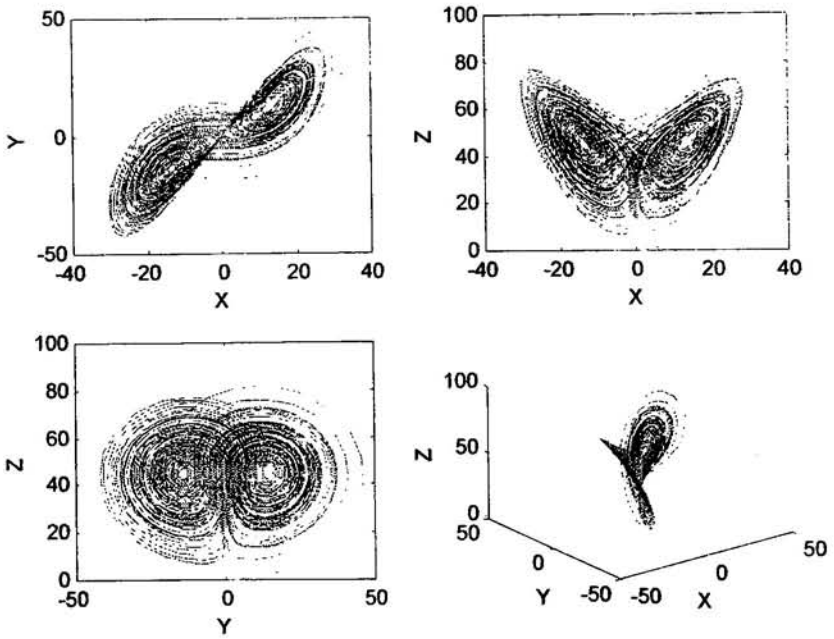


Figure 5.1: Projections of Lorenz Attractor

The Rössler equations are another set of three first-order differential equations. They were developed by Otto Rössler during his work with chemical kinetics. This set of equations only includes a single nonlinearity and as a result is considered weakly chaotic.

The three equations that compose the Rössler system are shown as Eq. 5.4, Eq. 5.5, and Eq. 5.6.

$$\frac{dx}{dt} = -y - z \tag{5.4}$$

$$\frac{dy}{dt} = x + ay \tag{5.5}$$

$$\frac{dz}{dt} = b + z(x - c) \tag{5.6}$$

The Rössler equations also depend on the three control parameters, a, b, and c, to determine their behavior. The most commonly studied set of values for these parameters are $a = 0.15$, $b = 0.20$, and $c = 10$. For these control parameters, the Rössler system will exhibit chaotic behavior that has been studied extensively. Published values are available for many of the analyses[27,52]. Four two-dimensional projections of the Rössler attractor for these parameters are displayed as Fig. 5.2.

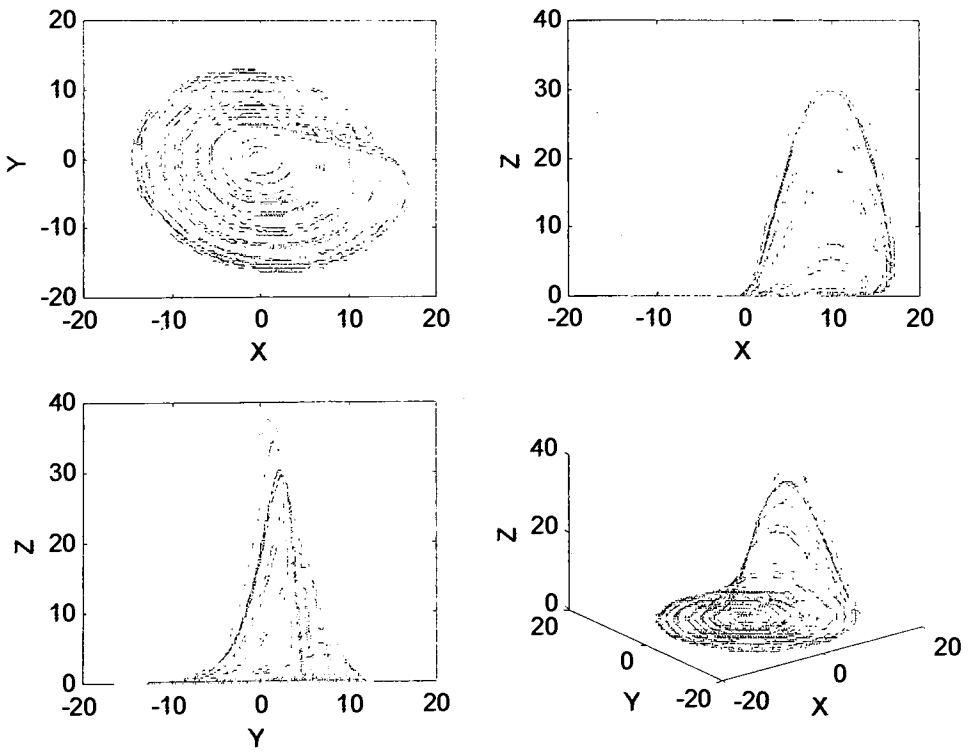


Figure 5.2: Projections of Rössler Attractor

The final nonlinear system that used in this study was the Duffing equation. The Duffing equation is a weakly nonlinear oscillator. When the Duffing equation is periodically forced, it can display chaotic behavior for various sets of parameter values. The form of the Duffing equation that was used in this study is shown as Eq. 5.7.

$$\frac{d^2x}{dt^2} + \epsilon\frac{dx}{dt} + x(x^2 - 1) = Fcos(\omega t) \tag{5.7}$$

Because this system is a second-order differential equation that is non-autonomous, it can be rearranged as three first-order differential equations. A **non-autonomous** system is a system whose dynamics are that is dependent on time. The three first-order differential equations that were used to study this system are shown below.

$$\frac{dx_1}{dt} = x_2 \tag{5.8}$$

$$\frac{dx_2}{dt} = Fcos(\omega x_3) - \epsilon x_2 - x_1(x_1^2 - 1) \tag{5.9}$$

$$\frac{dx_3}{dt} = 1 \tag{5.10}$$

Using these three first-order differential equations, the nonlinear behavior of the Duffing equation was simulated to demonstrate chaotic behavior. The three parameters used to control the behavior of the system are F, ω, and ϵ. The values used for these parameters where $F = 0.3$, $\omega = 1$, and $\epsilon = 0.25$. Figure 5.3 includes four two-dimensional projections of the attractor created by the Duffing equation when these parameter values are used. Because one of the state variables is time, two of the projections are the time series of x and $\frac{dx}{dt}$.

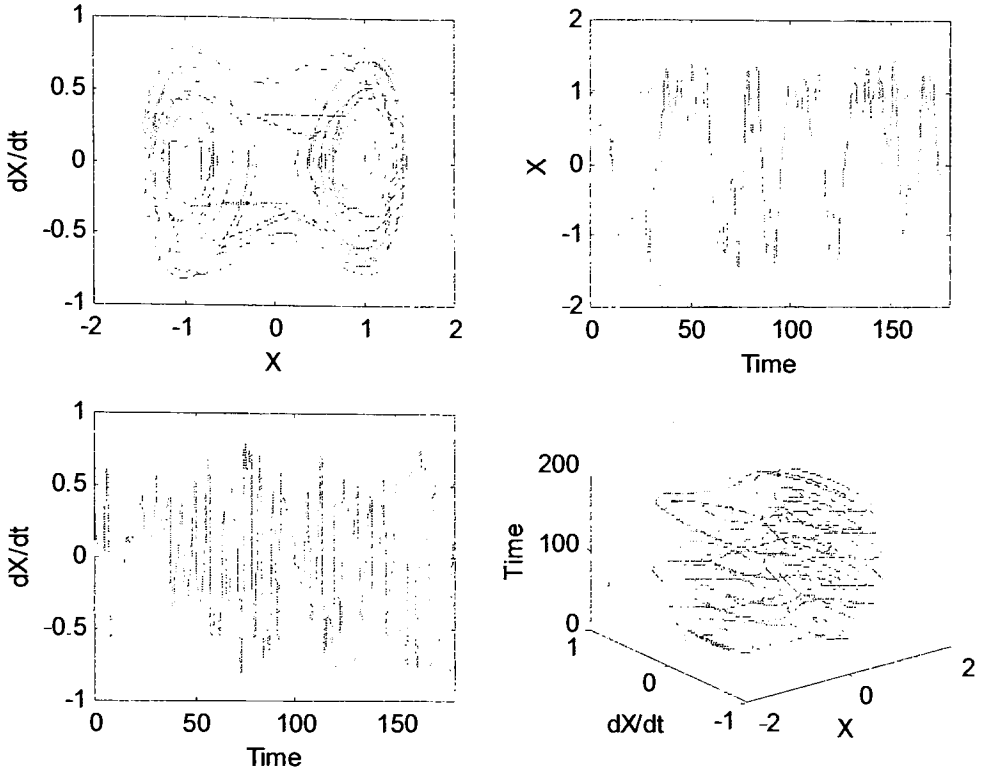


Figure 5.3: Projections of Duffing Attractor

Due to the nature of chaotic systems and the exponential divergence of nearby trajectories, great precision is required in the numerical integration of each system. As a result, the common methods for numerical integration such as Euler's Method and the Runge-Kutta Method were not employed. The three nonlinear systems were numerically integrated using a Lie Series approximation.

By constructing a Lie Series approximation for each nonlinear system, it was possible to develop a set of difference equations with variable coefficients and arbitrary initial conditions for each of the systems[50]. The nature of the of the Lie Series approximation also allows for the order of the approximation to be determined by the user. As the order of the approximation, n, is increased, the length of the difference equations increase exponentially but the error in the approximation is of the order of magnitude of the time step, h, to the power of the order of the approximation.

$$Error \sim O(h^n) \tag{5.11}$$

Due to the growing length of the difference equations and the capabilities of the available MATLAB version, a fifth-order approximation ($n = 5$) of each system was used for this study. The key component of the Lie Series approximation is the characteristic infinitesimal generator[47]. The characteristic infinitesimal generator is an operator that utilizes the differential equations and derivatives to determine the solution to a system of differential equations. The formula for the characteristic infinitesimal generator is shown as Eq. 5.12.

$$U = F_1\frac{\partial}{\partial x_1} + F_2\frac{\partial}{\partial x_2} + \ldots + F_N\frac{\partial}{\partial x_N} \tag{5.12}$$

In this equation, F_i is the time derivative of the ith component of the vector $\vec{x}$. The value of F_i can be a function of multiple components of $\vec{x}$ and is represented by the following equations.

$$\frac{dx_1}{dt} = F_1(\vec{x}) \tag{5.13}$$

$$\frac{dx_2}{dt} = F_2(\vec{x}) \tag{5.14}$$

$$\vdots$$

$$\frac{dx_N}{dt} = F_N(\vec{x}) \tag{5.15}$$

With the aid of this operator, the Lie Series can be used to determine the solution of a differential equation or set of differential equations. The formula for the Lie Series of a given differential equation is displayed as Eq. 5.16[46].

$$x_{s+1} = \sum_{k=0}^{\infty}\frac{h^k}{k!}(U^k x_s) \tag{5.16}$$

Since determining the exact solution of most nonlinear systems is not possible, the summation in this equation is only taken from zero to an integer value so that an approximation of the solution can be determined. The expansion of the formula is shown below.

$$x_{s+1} = x_s + h(Ux_s) + \frac{h^2}{2!}(U^2x_s) + \frac{h^3}{3!}(U^3x_s) + \dots \tag{5.17}$$

When the characteristic infinitesimal generator appears with an exponent, it is operating on the variable multiple times. Calculation of the third-order Lie Series approximation of a simple nonlinear equation is presented to provide an example of this procedure. Consider this initial value problem $\frac{dx}{dt} = \alpha x^2$. Here $F(x) = \alpha x^2$. The infinitesimal generator for this problem is given by the operator:

$$U = \alpha x^2 \frac{\partial}{\partial x} \tag{5.18}$$

Application of this operator to the variable x produces the following results.

$$Ux = F\frac{\partial}{\partial x}(x) = \alpha x^2 \tag{5.19}$$

$$U^2x = U(\alpha x^2) = F\frac{\partial}{\partial x}(\alpha x^2) = (\alpha x^2)(2\alpha x) \tag{5.20}$$

$$U^3x = U(2\alpha^2 x^3) = (\alpha x^2)(6\alpha^2 x^2) \tag{5.21}$$

The Lie Series of the solution given the initial condition of x_0, is:

$$x(dt = h) = x_0 + h(\alpha x_0^2) + \frac{h^2}{2!}(2\alpha^2 x_0^3) + \frac{h^3}{3!}(6\alpha^3 x_0^4) + \dots \tag{5.22}$$

Hence the third-order approximation is given by:

$$x(dt = h) = x_0 + h(\alpha x_0^2) + \frac{h^2}{2!}(2\alpha^2 x_0^3) + \frac{h^3}{3!}(6\alpha^3 x_0^4) \tag{5.23}$$

Using Eq. 5.23, it is possible to numerically integrate an approximation of the solution to this initial value problem for any value of α and any initial value of x_0. For

higher accuracy, the order of the approximation can be increased or the size of the time step can be decreased.

5.3 LABORATORY EQUIPMENT

After all of the algorithms were tested against published values and then used to determine how efficient each of the reconstruction methods were, the prevailing methods for attractor reconstruction were used on data collected from actual physical systems. Multiple data sets were collected from two of the three nonlinear devices capable of displaying chaotic behavior owned by the Mechanical Engineering Department at the Rochester Institute of Technology. The two devices from which data was collected were the Chua's Circuit device and the Multi-well Oscillator device.

The Chua's Circuit device is a nonlinear circuit composed of resistors, capacitors, operational amplifiers, and an inductor. Figure 5.4 shows the schematic for the circuit. The nonlinearity of the circuit comes from an arrangement of the operational amplifiers and resistors called Chua's Diode.

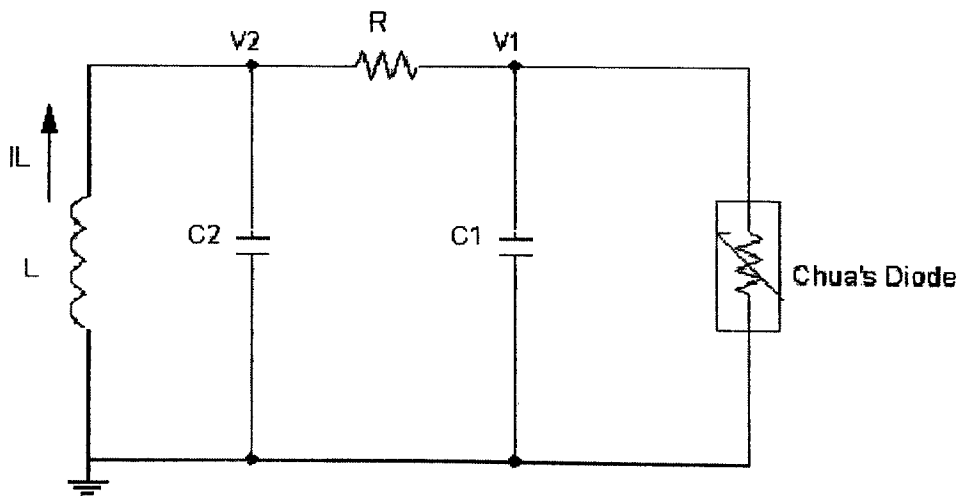


Figure 5.4: Schematic of Chua's Circuit

The arrangement of the operational amplifiers in Chua's Diode results in a nonlinear resistance profile. Figure 5.5 shows an example of the nonlinear resistance profile.

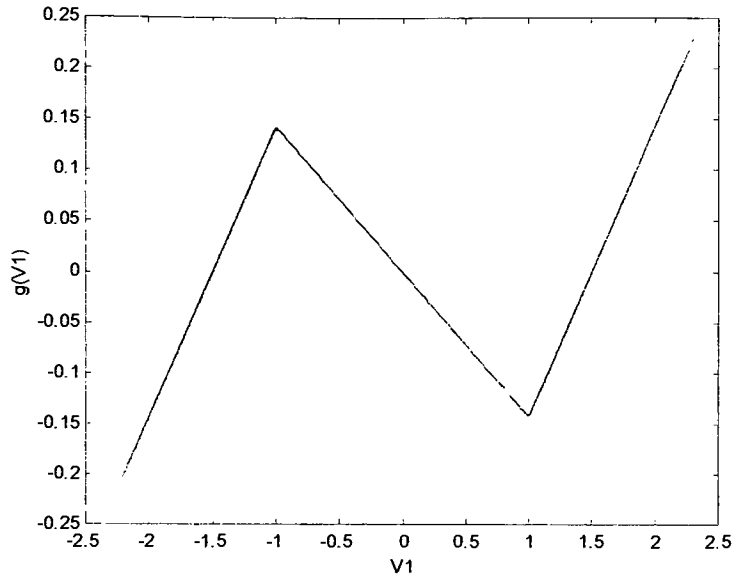


Figure 5.5: Nonlinear Resistance Profile of Chua's Diode

When the circuit is analyzed, three differential equations are derived with the voltage at two locations on the circuit and the current through the inductor as the three state variables. The equations for this circuit are shown as Eq. 5.24, Eq. 5.25, and Eq. 5,26.

$$C_1 \frac{dV_1}{dt} = \frac{V_2 - V_1}{R} - g(V_1) \tag{5.24}$$

$$C_2 \frac{dV_2}{dt} = \frac{V_1 - V_2}{R} + I_L \tag{5.25}$$

$$L \frac{dI_L}{dt} = -V_2 \tag{5.26}$$

The design of the Chua's Circuit device allows both of the voltages to be measured. This provides a two-dimensional projection of the original attractor that can be used to validate the reconstruction. These equations were also used to simulate behavior of the circuit during the design and testing of the device. For a given set of

parameter values, this nonlinear system is capable of producing chaotic behavior. Simulation of the system produced the two-dimensional projections of the attractor seen in Fig. 5.6.

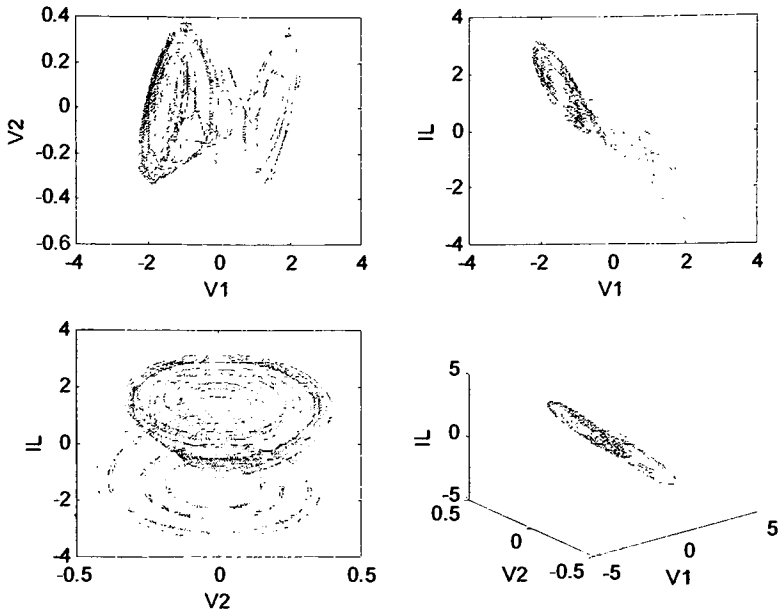


Figure 5.6: Projections of Chua's Circuit Attractor

The other physical system from which data was collected is the Multi-well Oscillator device. This piece of laboratory equipment is a magneto-elastic system that utilizes rare earth magnets and the nonlinear nature of buckling of a slender beam to create a nonlinear system capable of producing chaotic behavior. Figure 5.7 is an image of the CAD model used to develop the device.

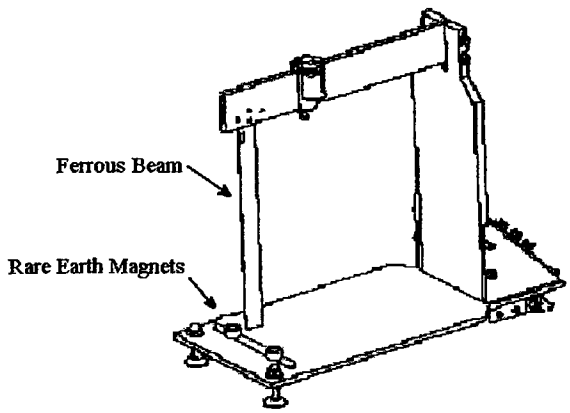


Figure 5.7: CAD Model of Multi-well Oscillator

In its Double-well Oscillator configuration, two rare earth magnets are positioned underneath a long, thin ferrous beam. The two magnets create a double-well potential that restricts the motion of the thin ferrous beam when a sinusoidal force is applied to the horizontal beam. Figure 5.8 shows an analog to the potential field where the magnets are located at horizontal positions of +1 and -1. When the applied force is weak, the beam will oscillate above one of the two magnets periodically. If the force applied to the beam is greatly increased, it will cause the beam to oscillate above both of the magnets in a periodic fashion and the magnets will no longer have a significant affect on its motion. The amount of energy added to the beam will be significantly greater than the energy required to move from the position of lowest potential to the area of highest potential between the two wells.

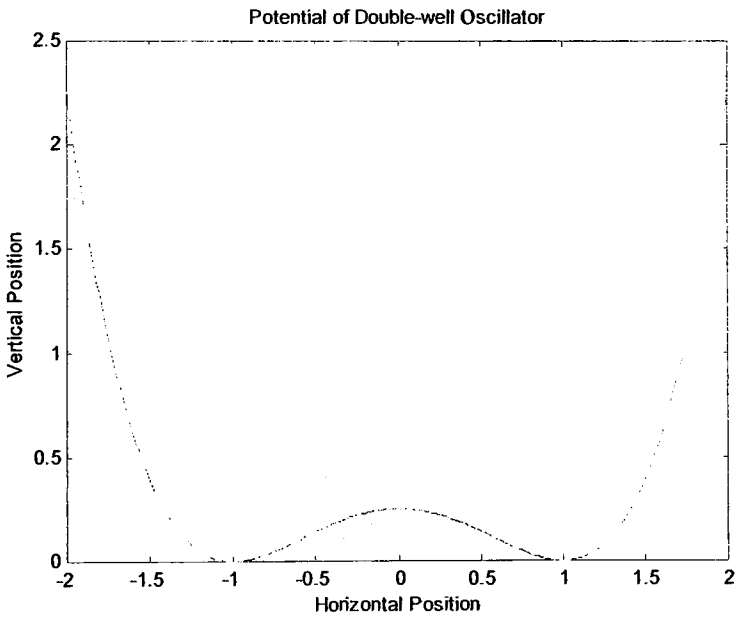


Figure 5.8: Double-Well Potential Analog Profile

When the strength of the applied force is at an intermediate level between these two extremes, the system will exhibit chaotic behavior. As the strength of the force is increased from its initial weak level, it will begin to move further into the area of higher potential between the two magnets. When just enough force is applied to the beam that it

is able to reach the position of highest potential between the two magentas, it will begin to move between the two magnets erratically.

The behavior of the Double-well Oscillator is governed by the Duffing equation. The x^3 term in the equation produces the double-well potential that constrains the motion of the beam. In the case of the Double-well Oscillator, the time series of the x variable can be easily interpreted. Figure 5.9 shows the time series of the x variable of the Duffing equation. The time series shows that the beam oscillates about one of the magnets then moves to a position above the other magnet where is oscillates for a non-uniform amount of time. Examining the entire time series, it can be seen that the beam moves between the two magnets in an unpredictable fashion.

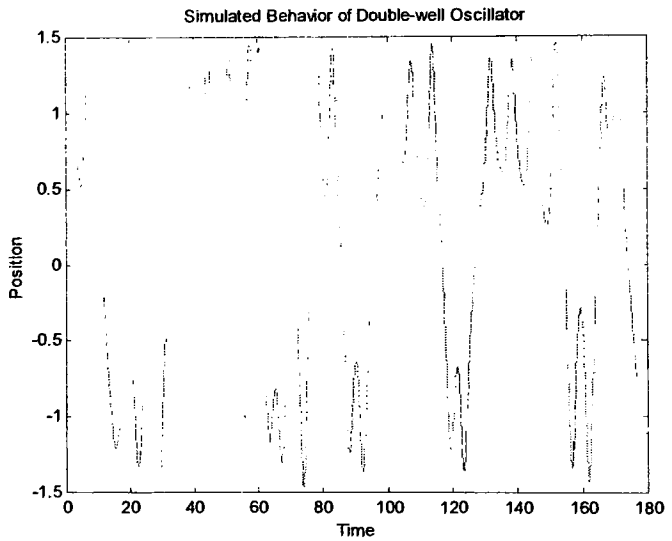


Figure 5.9: Time Series of the Beam Position of a Double-well Oscillator

5.4 COMPUTATION

The algorithms to determine the Lyapunov exponents and the various types of fractal dimensions of these data sets were implemented into scripts using The MathWorks, Inc.'s MATLAB software package, version 5.3. MATLAB is a tool for performing numerical computations with matrices and vectors. It also has the capability to run script files using an interpreted programming language. Each of the algorithms to

create and analyze the data sets was implemented into MATLAB scripts, also referred to as M-files. Some of these algorithms were then declared as functions and could be invoked from other scripts.

The calculations required to develop the Lie Series approximations of each of the nonlinear systems were done using Waterloo Maple, Inc.'s Maple 7 software package. Maple is a symbolic computation program that enables the user to perform calculations with arbitrary variables. The equations for each of the coefficients in the Lie Series approximation and their derivatives were calculated using Maple and then incorporated into the MATLAB M-files.

The majority of the work and calculations of this study were done using an IBM compatible computer with an Intel Pentium 4 processor operating at 2.40 GHz. The system had 512 MB of RAM and was running Windows XP. Other computers used during this study were IBM compatible computers with Intel Pentium 3 processors operating at about 700 MHZ. These systems were equipped with 256 MB of RAM and were running Windows 98 and Windows NT. Over 150 hours of computer time were required to complete the calculations reported in this study.

6 SOFTWARE IMPLEMENTATION

6.1 GENERAL

After each algorithm was studied, it was necessary to implement them using Maple and MATLAB so that data sets could be created and analyzed. The Lie Series approximations of each system were calculated using a Maple Worksheet based on work done by Andrey Vasilik and Dr. Josef Török[50]. The Worksheet determined approximations of the system's solution as well as the Jacobian matrix in terms of variable parameters and arbitrary initial conditions. The iteration of each data set and all of the analyses that where performed on the data sets were developed in MATLAB. The implementation of the algorithms was done by following the theory of the analysis method and examining other methods of implementation found in the resources. Copies of each of the programs used in this study appear in the appendices of this thesis.

6.2 MAPLE

The three first-order differential equations that govern the dynamics of each system were entered into the Worksheet. The Worksheet first placed the three state variables into two row vectors, $vars$ and w, and the three first-order differential equations into another, vec. The Worksheet then established variables as the initial values for each of the state variables. Once the necessary variables and vectors were created, the Worksheet was ready to iterate an Nth order approximation of the system solution. The iteration started by calculating the Jacobian matrix, $[J]$, of w with respect to $vars$.

$$\{vars\} = [x \quad y \quad z] \quad \{vec\} = [F_1 \quad F_2 \quad F_3] \quad \{w\} = [x \quad y \quad z]$$

$$[J] = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{bmatrix} * \{w\}^T \tag{6.1}$$

The Jacobian matrix of the first iteration was the identity matrix. After the Jacobian matrix was calculated, it was multiplied by the *vec* vector to produce a three element vector.

$$[J]\{vec\}^T = [U^k x \quad U^k y \quad U^k z] \tag{6.2}$$

The three components of this vector are the characteristic infinitesimal generator operators for the kth iteration of the 'for' loop. After these values are determined, the Worksheet multiplies them by h to the kth power and divides by the factorial of k. Following this process in the iteration, the row vector of characteristic infinitesimal generator operators becomes the new w vector and the process is repeated.

$$\{w\} = [U^{k-1} x \quad U^{k-1} y \quad U^{k-1} z]$$

When the Nth iteration is complete, the Worksheet presents the Nth order approximation of the solution to the system represented by the three first-order differential equations. The final Jacobian matrix produced by the Worksheet is the Nth order approximation of the change in each of the differential equations caused by a change in each variable.

After the approximations and Jacobian matrices for each of the systems were calculated using Maple, they were incorporated into MATLAB M-files so that the systems could be simulated and data sets could be created to use in the analysis.

6.3 MATLAB

6.3.1 SERIES ITERATION

After the Lie Series approximations were calculated using the Maple Worksheet, they were transferred into a MATLAB M-file. The solutions for each of the systems were initially calculated using a 7th order approximation but after initial testing of the script files, it was determined that the memory capabilities of MATLAB could only handle a 5th order approximation. Because the parameters and state variables remain unassigned the length of the difference equations increases exponentially as the order of the approximation is increased. In a study of the Lie Series approximations done by Andrey Vasilik, he stated that a fifth-order Lie Series approximation will produce a solution with less error than the standard fourth-order Runge-Kutta approximation[50].

The script file was written such that each of the terms from the characteristic infinitesimal generator were positioned separately. This allowed the function to be written such that one of the input parameters is the order of the approximation from first-order to fifth-order. Preliminary examination of the different order approximations indicated that the lower order approximations did not produce an accurate enough estimation of the solution and the fifth-order approximation did not require significantly more time than the others. As a result, the fifth-order approximation was used to create all the simulated data sets that were used in the study.

The script file containing the Lie Series approximations was developed so that the inputs were the time step, the current point in state space, the order of the approximation and the name of the system to iterate. Using this information, the M-file would utilize the Lie Series approximation for the desired system and order. Using the inputted time step and location in state space, the function calculates the next point in state space. The function also calls upon one of the other script files to determine the Jacobian matrix for

the system and location being examined. The output of this function is the next position in state space and the Jacobian matrix.

Another script was written that would call upon the M-file containing the Lie Series approximations within a 'for' loop so that a time series of the state variables could be created. To verify that the data set was created properly, a series of figures were plotted showing the time series of each of the state variables as well as various projections of the system's attractor. As with most of the other script files, code was added so that the date, time, and length of time required for the iterations would also be displayed.

6.3.2 LYAPUNOV SPECTRUM

The first analysis performed was done to determine the spectrum of Lyapunov exponents for each of the chaotic systems that were being studied. To verify that the algorithm developed for the analysis was functioning correctly, a fifth-order Lie Series approximation was calculated for a simple linear system. Because the system is linear, the rate of exponential convergence can be determined analytically. The linear system used is shown as Eq. 6.3.

$$\frac{d^2x}{dt^2} + 3\frac{dx}{dt} + 2x = 0 \tag{6.3}$$

The variable x in the equation is substituted with the function $x = e^{rt}$ and then solved for the value of r.

$$r^2 + 3r + 2 = 0 \qquad r = -1, -2 \tag{6.4}$$

To maintain the format of a three-dimensional system, a third differential equation was added to account for time. Analysis of the function to determine the Lyapunov Spectrum was successfully able to determine the three exponents to be 0, -1, and -2 for a logarithmic base of $e = 2.7183...$ When the Lyapunov exponents are

calculated for the nonlinear systems, a logarithmic base of two was used so that the exponents have the standard units of bits per second.

The first section of the function to determine the Lyapunov Spectrum for a system iterates the system from a set of initial conditions until the trajectory of the system has reached the attractor. To verify that the trajectory has successfully converged to the attractor, a three-dimensional plot of the trajectory is produced.

After the trajectory has been iterated past the transient portion, the process to determine the Lyapunov exponents begins. The first step is to establish three orthogonal vectors that can serve as a basis for the space. The initial basis vectors arranged as the rows of matrix E cause the matrix to resemble the identity matrix. This matrix is then multiplied by the transpose of the Jacobian matrix produced by the functions performing the Lie Series approximation. The Jacobian matrix is transposed so that the columns of the matrix will represent the change in each of the difference equations caused by a change in each of the variables. The product of these two matrices represents the growth and decay of the basis vectors within state space. Equation 6.5 displays the formula for this procedure.

$$[V] = [E] * \begin{bmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_2}{\partial x} & \frac{\partial F_3}{\partial x} \\ \frac{\partial F_1}{\partial y} & \frac{\partial F_2}{\partial y} & \frac{\partial F_3}{\partial y} \\ \frac{\partial F_1}{\partial z} & \frac{\partial F_2}{\partial z} & \frac{\partial F_3}{\partial z} \end{bmatrix} \tag{6.5}$$

The new row vectors of $[V]$ must now be normalized and orthogonalized. The first step in this process is to determine the magnitude of the first row. The magnitude is calculated in the standard Euclidean manner. Each element of the first row of matrix $[V]$ is then divided by its magnitude in order to normalize the vector. Once the first vector is normalized, the other two vectors must be orthogonalized. This orthogonalization is done using the Gram-Schmidt method explained in Chapter Two. During the orthogonalization process, the magnitudes of all the row vectors of matrix $[V]$ are

determined and following the orthonormalization, they are used to determine the Lyapunov exponents for the attractor. The following equation displays the method employed to calculate the three Lyapunov exponents for each attractor.

$$\lambda_i = \frac{1}{N*dt}\sum_{i=1}^{N}\frac{log(mag_i)}{log(base)} \tag{6.6}$$

This equation displays the formula used to calculate the ith Lyapunov exponent, λ_i, from all the magnitudes, mag_i, measured of the ith vector. The natural logarithm of this value is determined and then divided by the natural logarithm of the desired logarithmic base, $base$. This allows the function to compute the exponents with a base of e for the linear system and a base of two for the nonlinear systems. After the sum of all the logarithms are calculated, the total is divided by the total amount of time that passed as the trajectory evolved. This value is calculated by multiplying the number of cycles examined, N, by the time step used, dt. An additional calculation was added that allowed the value of the exponent to be calculated following each cycle. This data set was then plotted so it could be used to verify that the calculation of the Lyapunov exponents had been evolved long enough to converge to the correct values. Figure 6.1 is the output plot created by this function showing the convergence of the exponents to the correct values for the linear system.

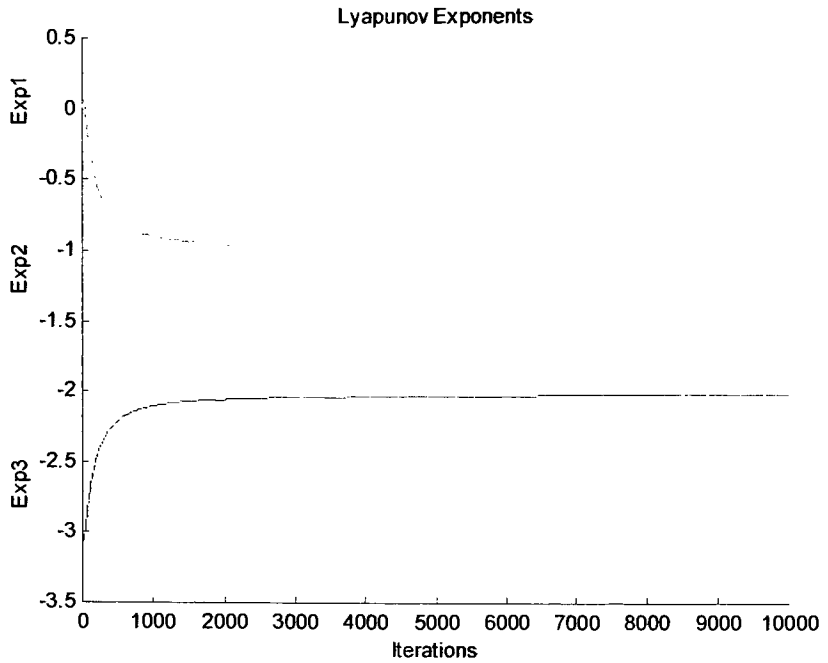


Figure 6.1: Convergence of Lyapunov Exponents of Linear System

Since this function is able to calculate all of the Lyapunov exponents, it is able to use these values to calculate the Lyapunov Dimension for the chaotic systems. Calculation of the Lyapunov Dimension is done using the formula given in Eq. 4.17 in Chapter Four.

6.3.3 LARGEST LYAPUNOV EXPONENT

The purpose of this function is to examine a data set and from its trajectory, determine the largest Lyapunov exponent that would describe the behavior of the attractor. The basic design of this function follows the theory presented in Chapter Two in section 2.2. The function starts by omitting the beginning of the data set so as to avoid the transient period of the trajectory converging to the attractor. The transient period was determined visually. The function then creates a subset of data from the original time series. The next step was to determine the distance between the first point in the data set, the subject point, and all the other data points within the data set. Because this was done for each cycle, a subset of the original time series was used to reduce the time required

for computation. The function also takes measures so that the data point selected to be the closest was not on the same orbit of the trajectory as the subject point and not at the very end of the trajectory. The trajectory was then followed through the time series for both data points to determine the distance between both points after a predetermined amount of time passed. This amount of time was chosen to be large so that the errors caused by orientation would minimally affect the results. The final distance between the two points was then divided by the initial distance to determine the amount of growth or decay that occurred. The data point to which the previous subject point was traced to along the trajectory becomes the new subject point. Once again, the remainder of the data set is examined to determine the closed data point, not on the same orbit of the trajectory. The two points are again followed along the trajectory, distances are determined and the process continues for the number of cycles inputted into the function.

As with the previous function, the natural logarithm of the ratio between the final and initial distances was taken and divided by the natural logarithm of the desired base. The sum was then updated following each cycle of the function. Following the last cycle, the sum was divided by the total amount of time of the data set that was examined. In this function, the total amount of time that passed was equal to the product of the number of cycles, the number of time increments along the trajectory that the two points are traced, and the time step of the data set. As with the previous function, additional code was added to monitor the value of the Lyapunov exponent during the iterations. Figure 6.2 shows the convergence of the largest Lyapunov exponent for the Lorenz system with a characteristic set of parameter values.

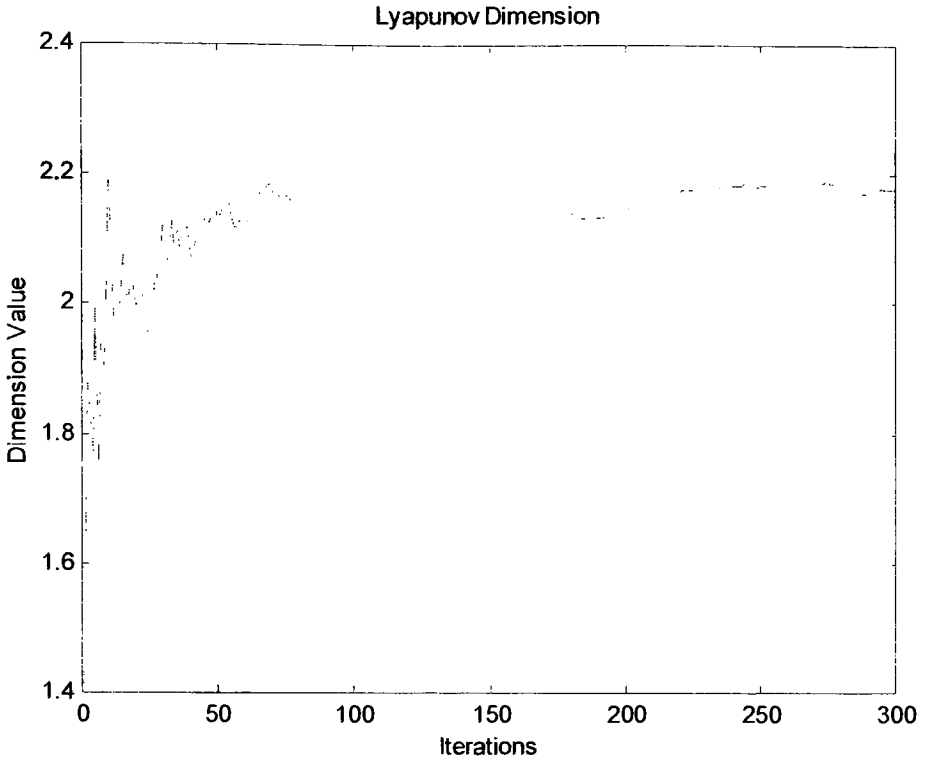


Figure 6.2: Determination of Largest Lyapunov Exponent

Since it is necessary to follow the data points along the trajectory for a relatively large number of time increments, a large number of cycles is required to allow the Lyapunov exponent to converge. Consequently a large number of data points are required. Combined with the portion of the function determining the distance between the subject point and the rest of the data points in each cycle, this function requires a large amount of computation time to produce accurate values.

6.3.4 LINEAR REGRESSION

This function, based on the work of William Robertson and Dr. Josef Török[37], utilizes a simple linear regression process to fit the data sets inputted to a straight line. The process begins with the two data sets; the independent variable and the dependent variable. The function also incorporates the indices used to determine on which subsets of these data sets the linear regression is to be applied. Once the subsets are created as

column vectors, the function creates a two column matrix where the first column is the subset of independent variables and all the components in the second column have a value of one. The next step in the linear regression process calls for the transpose of this two column vector to be multiplied by itself and the column vector of the dependent variable values. This produces a two element by two element matrix designated $[A]$ and a two element column vector designated $\{B\}$. The calculation of these two terms are shown in Eq. 6.7 and Eq. 6.8.

$$[A] = \begin{bmatrix} x_1 & x_2 & \dots & x_N \\ 1 & 1 & \dots & 1 \end{bmatrix} * \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \tag{6.7}$$

$$\{B\} = \begin{bmatrix} x_1 & x_2 & \dots & x_N \\ 1 & 1 & \dots & 1 \end{bmatrix} * \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{6.8}$$

After these two terms have been determined, the slope and intercept for the line that best fits the curve can be determined. Because of the relationship between matrix $[A]$ and vector $\{B\}$, the inverse of $[A]$ is multiplied by $\{B\}$ to produce these values. Equation 6.9 and Eq. 6.10 shows the relationship between these terms and how the slope, m, and intercept, b, are determined.

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \tag{6.9}$$

$$\begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} * \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \tag{6.10}$$

After the slope and intercept for the line that best fit the subset of data are determined, the function plots the entire data set as well as the fit line for the region over which it was applied. To ensure that the function was operating properly, it was tested with linear data sets. Figure 6.3 is an example of the output plot created by the M-file.

Although the entire data set is linear, the linear regression process was only conducted on a portion of the data to verify the functions ability to examine subsets.

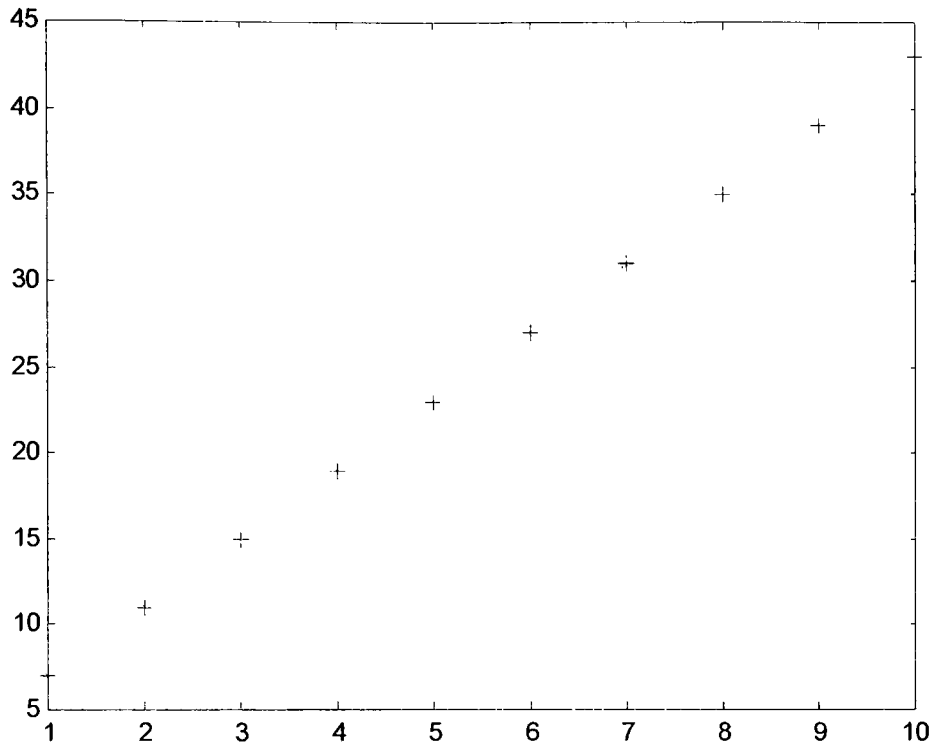


Figure 6.3: Testing Linear Regression Function on Linear Data

The function also calculates the square of the Pearson product moment correlation coefficient. This value represents the proportion of the variance in the dependent variable attributed to the variance in the independent variable. When this value is equal to one, the line perfectly fits the data. Values less than one indicate that the data is not perfectly linear. The equation for determining the Pearson product moment correlation coefficient is presented as Eq. 6.11.

$$r = \frac{n(\sum XY) - (\sum X)(\sum Y)}{\sqrt{[n\sum X^2 - (\sum Y)^2][n\sum Y^2 - (\sum X)^2]}} \tag{6.11}$$

In this equation, the variable n represents the length of the data set, X represents data set of independent variables, Y represents the data set of dependent variables. The

linear regression function is called upon by each of the function to determine the fractal dimensions of the strange attractors. This is done following the procedure presented in Chapter Four and best displayed in Fig. 4.3. The calculation of the 'R-Squared' term is used to verify that the linear regression is performed on the most linear portion of the curve.

6.3.5 CAPACITY DIMENSION

In order to calculate the Capacity Dimension for an attractor, two important things must be done. First the area occupied by the attractor must be divided into equal sized volume elements. Then each volume element must be examined to determine if any portion of the attractor is present within the element. When the number of elements occupied by the attractor has been determined for a number of different element sizes, it is possible to determine the Capacity Dimension of the attractor.

As with all the other functions, this function starts by creating a subset of the data set skipping the transient period and using a smaller number of data points to allow for shorter computation time. To accomplish the first step in this process, the time series of each state variable is examined to determine the maximum and minimum values. This provides the function with the region in state space where the attractor is located. The minimum values are rounded down and the maximum values are rounded up. This ensures that a data point will not exist on the boarder of the area under examination. The lengths used to create the volume elements are determined by the function based on the size of the attractor calculated from the maximum and minimum values. The function then determines the necessary element lengths so that they will be evenly spaced when plotted on a logarithmic scale.

The function then cycles through each of the different element sizes and creates an array of volume elements that cover the entire strange attractor. By maintaining a record of the maximum and minimum values for all three directions of each volume

element, the function can cycle through the entire data set and determine if any of the points fall within the bounds of the volume element being examined. Because this dimension is only concerned with the number of volume elements occupied and not the number of data points within each element, after a point is located within a volume element, the function stops examining the data set and moves on to the next volume element.

The function is designed to start at the volume element that coincides with the minima of the three variables. The opposite side of the volume element was determined by adding the length of the element to these values. After the function establishes the boundaries of the volume element, and has searched through the data set to determine if any data points are present, the boundaries of the next volume element are calculated. This is done by using the maximum value face on the first axis as the new minimum value and adding the element length to this value to determine the new maximum value of the volume element. When the maximum value face of the volume element exceeds the maximum value of the variable along the first axis, the initial values for the first axis of the volume element are employed to return the element to the position at the minimum of the variable. At the same time, the two faces of the volume element normal to the second axis are moved so that the minimum face exists at the value that previously was the maximum face and the value of the maximum face in increased by one element length. When the first and second axes have completely covered a two-dimensional area of the attractor, they are reset to their initial values and the values for the faces of the volume element normal to the third axis are increased in a similar fashion. This process is continued until the an array of volume elements has been constructed to examine the entire area inhabited by the data set. After the boundaries have been established for each volume element, the function searched through the data set to locate any data points that fall within the confined area. To ensure that no data points that fall between volume elements are omitted, the function searches for points greater than the minimum values

and less than or equal to the maximum values. When all six of these conditions are met, the function ceases to examine the data set, adds one to the number of inhabited volume elements, and moves to the next volume element.

When the function has examined the attractor using all the different length volume elements, it is almost ready to call upon the linear regression function to determine the fractal dimension. The last step before this can be completed is to determine which subset of the values containing the number of cubes and element lengths is to be examined by the linear regression function. This is done by examining the data set of the number of cubes and determining which elements have a value greater than one. When the number of elements is equal to one, the size of the volume element was so large that the entire attractor fit within one volume element. Finally, the linear regression function is applied to the natural logarithm of the number of cubes and the natural logarithm of the element lengths. The slope of the curve determined by the linear regression function is the Capacity Dimension of the attractor.

6.3.6 INFORMATION DIMENSION

The theory behind the Information Dimension causes the format of this function to be very similar to the function to determine the Capacity Dimension of an attractor. As with the previous function, this function must also cover the entire attractor with volume elements of various lengths. The difference between the Capacity Dimension and the Information Dimension is that the Information Dimension is concerned with how many points fall within each of the volume elements.

The function to determine the Information Dimension starts out similar to the previous function. It determines the maximum and minimum values of each state variable to determine the volume of state space that must be determined. The maxima are rounded up and the minima are rounded down. The function then increments its way

through the first axis, incrementing the second axis when the first has traversed the length of the attractor. When the second axis has reached its maximum, the function moves to the next position along the third axis. This process continues until the entire attractor has been covered with volume elements.

As each volume element is being examined, the function scrolls through the subset of the data that is being examined and determines how many of the data points are within the bounds of the volume element. The number of points within the volume element is then divided by the total number of data points to find the probability that a data point will fall within the volume element being examined. This probability is used to calculate the entropy for the volume element which is added to the entropy for all of the other volume elements. The entropy of the attractor is calculated using Eq. 4.9 in Chapter Four.

After the sum of the entropy for all of the inhabited volume elements has been calculated for a number of different element sizes, the function determines which subset of data is to be used by the linear regression function. This is done by determining which element sizes produce entropy values greater than a level determined experimentally. The linear regression function is then used to determine the slope of the linear portion of the curve of the entropy versus the natural logarithm of the element sizes which is the Information Dimension.

6.3.7 CORRELATION DIMENSION

The most efficient method for determining the dimension of a data set is by calculating the Correlation Dimension. Because calculation of the Correlation Dimension does not require segmenting off the entire portion of state space occupied by the attractor into volume elements the amount of time required is significantly less than for the other methods. This characteristic also makes it possible to use a shortcut that further decreases the length of time required to determine the fractal dimension.

The first step in calculating the Correlation Dimension is to calculate the Correlation Sum. This is done by determining the average number of points within a given range of distances from each points. The shortcut used to hasten this process enables only a subset of M randomly selected points to be examined in order to determine how many data points are within each distance from them. The indices, i, for this subset of points are determined using a random number generator function and then scaling them to an integer value between zero and the total number of points in the data set. In the event that an index of zero is chosen, the function will reselect another value at random until a non-zero value is chosen.

Utilization of the Theiler coefficient, W, is accomplished by omitting a given number of data points immediately around the point being examined[43]. To ensure that this works properly, the randomly selected indices are not permitted to be within this number of data points from either end of the data set. In the event that the randomly selected index is equal to one of these values, a new index value is selected at random. When the randomly selected point is successfully determined, two summations are used to examining the data points on either side of this point within the data set.

$$C(r,N,M,W) = \frac{\left(\sum_{j=1}^{i-W} \theta(r - ||x_i - x_j||) + \sum_{j=i+W}^{N} \theta(r - ||x_i - x_j||)\right)}{(N - 2W + 1)M} \tag{6.12}$$

As with each of the previous two methods for determining the fractal dimension, after the Correlation Sum has been calculated for various element lengths, r, the two data sets are plotted against each other on a logarithmic-logarithmic graph and the linear portion of the plot is used to determine the fractal dimension. After performing numerous experiments, the subsets used by the linear regression function are selected by examining the values of the natural logarithm of the Correlation Sums. The slope of the this linear portion is the Correlation Dimension of the data set.

6.3.8 TIME DELAY FROM AUTOCORRELATION

This function was created to determine the various time delay values used for attractor reconstruction from the autocorrelation of the signal. This basic function starts with a data set for a single variable and computes the autocorrelation. After the autocorrelation of the data set has been calculated, this function examines the autocorrelation to locate the time delay values corresponding to positions including the first zero, first local minimum, first inflection point, and different fractions of the maximum value. When each of these values has been determined, the function plots the autocorrelation function along with the points where each of these values occur. This allows the user to verify that the function is working properly and that the points do coincide with their correct location on the autocorrelation of the data. The function also uses each of the delay values to create a two-dimensional projection of the reconstructed attractor. For extreme cases, inspection of the projection can be used to omit some delay values from further examination. The plot produced displaying the different delay value choices appears in the for of Fig. 6.4.

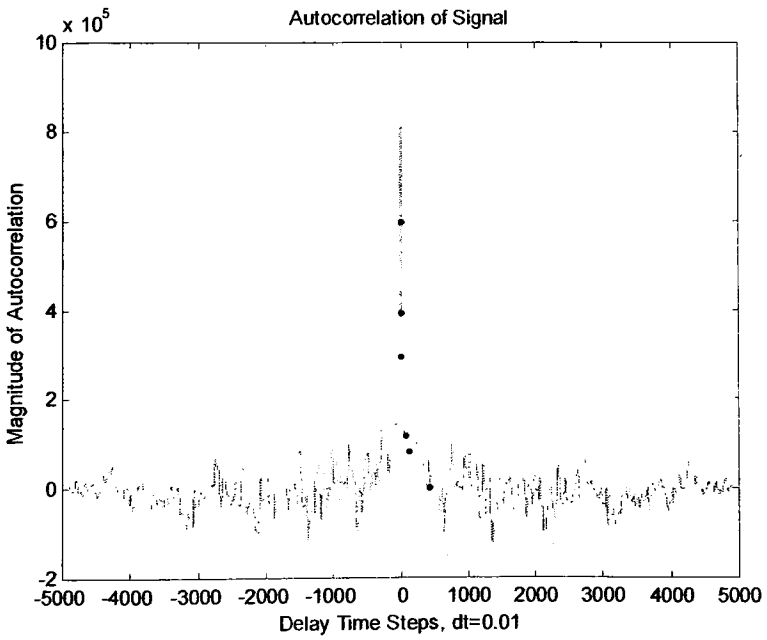


Figure 6.4: Autocorrelation of Function with Delay Values

The most important output from this function is the delay time values which are given in with units of seconds. Once these values have been calculated, it is very easy reconstruct the attractor. Using Eq. 3.2, the only other information required to reconstruct the attractor is the embedding dimension.

6.3.9 EMBEDDING DIMENSION

The function that determines the embedding dimension is very similar to that function that calculates the Correlation Dimension. However, while the Correlation Dimension function varies the distance around the points being inspected, this function maintains a constant distance and varies the embedding dimension of the data set. This function also employs the shortcut of randomly selecting a subset of data points from the set and determining the average number of points within the given distance of them. After a range of embedding dimensions have been examined, the function produces a plot of the average number of points within the given distance and the embedding dimension of the data.

In order to determine the minimum required embedding dimension, the plot was examined to determine when the average number of points no longer decreases at such a drastic rate. While the average number of points continues to decrease with the increase of the embedding dimension, the rate at which the average number of points decreases diminishes. This affect can be seen in Fig. 6.5.

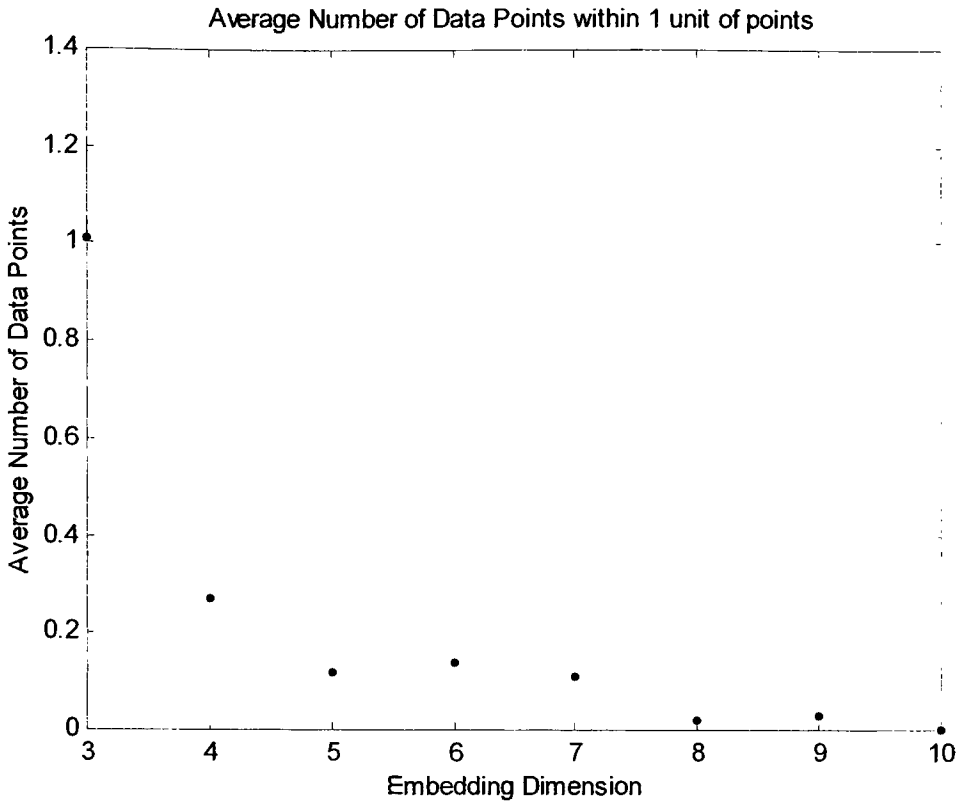


Figure 6.5: Average Number of Points Versus Embedding Dimension

Because of the wide use of the time delay value corresponding to the first inflection point of the autocorrelation function, it is used by this function to embed the data with the various embedding dimensions. From the plot in Fig 6.5, an embedding dimension of eight would be sufficient in create a high-dimensional space capable of adequately distributing the data points. This agrees with the statements made by Takens that the embedding dimension should be greater than twice the degrees of freedom of the original system plus one. With the embedding dimension, it is now possible to reconstruct the attractor using delay time values produced by the function that employed the autocorrelation function.

6.3.10 MUTUAL INFORMATION

After the initial preprocessing of the data to select a subset of the data and remove the transient period, this function embeds the data set into a two-dimensional space across a wide range of delay time values. These two component vectors can be plotted to produce a two-dimensional projection of the reconstructed attractor. After each delay time value has been used to reconstruct the attractor, the area inhabited by the projection is divided into equally sized squares. This is done by first determining the minimum and maximum values for each of the two elements. The minima are rounded down and the maxima are rounded up to ensure that no data points will fall on the boundary of the area under investigation. These minima and maxima are then used to determine the range of each element of the reconstructed vectors. The two ranges are divided by the number of segments desired to produce the necessary increment length.

Using the same methods to create a three-dimensional array of volume elements for the information dimension, this function creates a two-dimensional array of area elements. After the boundaries of each element are calculated, the function determines how many of the data points fall within the element. Once a two-dimensional matrix has been created so that it contains all of the data points of the projection, the number of data points in each area element is divided by the total number of data points to determine the probability that a point will fall within each of the area elements. The sum of all the probabilities in each column as well as each row can then be obtained from the matrix to produce the distribution of each separate element from the reconstructed vectors. These distributions are equivalent to histograms of the two elements. These probability distributions and the probability distribution matrix are then used with Eq. 3.14 to calculation the amount of information shared by the two components.

The amount of information is then plotted against the delay time value used to reconstruct the vector. The first local minimum of the amount of information shared by the two components is taken to be the location of the optimal delay time value. The

function plots the curve of information against the delay values and marks the location of the local minimum so that it can be verified by the user. The function then uses the delay time value to produce a two-dimensional projection of the reconstructed attractor.

6.3.11 AVERAGE DISPLACEMENT

The function to determine the Average Displacement follows the standard preprocessing methods as the other functions. The function starts by selecting a subset of the time series omitting the transient period where the trajectory approaches the attractor. Following the preprocessing of the time series, the average displacement is determined for a range of delay time values. This is done by calculating the difference between the first component of the reconstructed vector and all the other components. The number of components used in each reconstructed vector, the embedding dimension, is the value determined by the Embedding Dimension function. The differences are calculated directly from the time series, skipping the actual reconstruction of the attractor. Each of these distances are summed together and divided by the total number of reconstructed vectors.

After this process has been completed across a range of delay time values, the average displacement is plotted against the delay time values. Because the selection criteria for this method requires examination of the slope of this curve, it is necessary to use Eq. 3.16. Finally, the function locates the delay time value corresponding to when the magnitude of the slope falls below forty percent of the slope for a delay time value of one. From these calculations, the function produces the calculated delay time value, a plot of the average displacement versus delay time value, a plot of the slope of the average displacement versus delay time value, and a two-dimensional projection of the reconstructed attractor using optimal delay time value as determined by the function.

6.3.12 SINGULAR SYSTEM APPROACH

After completing the necessary preprocessing, the Singular System Approach function creates the trajectory matrix as displayed in Eq. 3.18. The number of components used to create the vectors in the trajectory matrix is calculated from one of the input variables. The delay window for these vectors is chosen so that the time between the first and last component of each vector is equal to the delay time corresponding to the first inflection point of the autocorrelation of the time series. After all the vectors have been established in the trajectory matrix, it is normalized by dividing it by the square root of the number of reconstructed vectors.

After completing the trajectory matrix, the process of singular value decomposition is done to determine the singular vectors and singular values. The process decomposes the matrix into three separate matrices as shown in Eq. 3.19. Because the $[C]$ matrix contains the singular vectors and can be used to obtain the singular values, use of the singular value decomposition function in MATLAB spends extra time determining unnecessary information. Instead of using MATLAB's function, the Singular System Approach function calculates the matrix product of the transpose of the trajectory matrix and itself and then determines the eigenvectors using another pre-existing MATLAB function. Once the singular vectors have been determined, the function calculates the singular values using Eq. 3.20.

After the singular values are determined, they are used to determine which of the singular vectors is the most dominant, as well as the order of dominance of the remaining vectors. Using the 'sort' function in MATLAB, the singular values are arranged into descending order. The index produced by the 'sort' function is used to arrange the singular vectors into the order corresponding to the decreasing singular values.

Now that the abridged version of singular value decomposition has been completed, the singular values must be further inspected to determine how many of the singular vectors are deterministically dominated and how many are noise dominated.

This is done by first calculating the sum of all the singular values and then determining the percent of that total that each singular values represents. The logarithm of base ten of these percentages are then plotted on a chart with the indices of the singular value as the independent variable. As stated in Chapter Three, visual investigation of this plot can yield the embedding dimension, d_e, of the reconstructed attractor.

The final step of this function is to multiply the trajectory matrix by the matrix of singular vectors to produce a matrix equal in size to the trajectory matrix but containing the projection of the original vectors onto the singular vectors. Reconstruction of the attractor can then be completed by using the first d_e columns of this new matrix.

7 ANALYSIS RESULTS

7.1 GENERAL

Through this study, the optimal methods for reconstructing an attractor from the time series of a single variable are determined. In order to accomplish this, the script files must first be validated to ensure that they are performing the analysis correctly. Because only a finite amount of data is to be examined, this testing will also indicate the accuracy capabilities of each function. While this testing is used to validate the application of each analysis into MATLAB, the testing also provides further verification that the Lie Series approximations are performing correctly.

After determining the Lyapunov exponents and various fractal dimensions of the complete data sets, one variable from the data set is selected. This single variable time series is then used to reconstruct the system's strange attractor using many of the methods discussed in Chapter Three. The reconstructed attractor is then used to determine the largest Lyapunov exponent as well as the fractal dimension of the data set. These two values are compared with the values obtained from the complete original attractor to determine which reconstruction method is the most successful.

The optimal reconstruction method is then applied to data collected from actual chaotic physical systems. This is done as a final verification that the process can be applied to systems where information for all the variables is not available.

7.2 FUNCTION VALIDATION

The first step in the analysis was to examine known data sets using each of the different functions to verify that they were performing properly and with adequate accuracy. Of the four systems that were examined, published values for the Lyapunov exponents, Lyapunov Dimension, and Correlation Dimension were found for the Rössler

system and one of the Lorenz systems. These values are presented in Table 7.1[27,48,52].

	Lorenz $\sigma = 16, r = 45.92, b = 4$	Rössler $a = 0.15, b = 0.20, c = 10$
Lyapunov Exponents	2.14, 0, -32.4	0.13, 0, -14.1
Lyapunov Dimension	2.06 ± 0.01	2.01 ± 0.01
Correlation Dimension	2.05 ± 0.01	2.01 ± 0.01

Table 7.1: Published Values for Studied Lorenz and Rössler Attractors

With these values available to verify the results of the different analyses, each of the different functions was thoroughly tested and enhanced to produce the most accurate results within a reasonable amount of time. Four simulations were examined using the function to determine the Lyapunov spectrum of a system in conjunction with a fifth-order Lie Series approximations for each system. As this was the first analysis conducted, each system was examined multiple times while varying different parameters. The different parameters varied were the sampling time and the length of time that each system was iterated. Data sets were created using sampling times of 0.01 and 0.001 seconds for ten thousand and one hundred thousand data points resulting in four different runs per system. The Lyapunov exponents calculated by each of these runs are presented in Table 7.2. In this table, the column labeled Lorenz-1 refers to the Lorenz system with the classical parameter values, $\sigma = 10$, $r = 28$, $b = 8/3$, and the column labeled Lorenz-2 refers to the Lorenz system with the popular parameter values, $\sigma = 16$, $r = 45.92$, $b = 4$. For the Rössler system and Duffing system, the parameter values defined in Chapter Five were used.

	Lorenz-1	Lorenz-2	Rössler	Duffing
Published Values	-	2.14	0.13	-
	-	0	0	-
	-	-32.4	-14.1	-
$T = 10\,sec$	0.791	1.970	0.166	0.046
$dt = 0.001$	0.281	0.113	0.019	0.000
	-20.79	-32.38	-11.67	-0.407
$T = 100\,sec$	1.287	2.139	0.136	0.153
$dt = 0.001$	0.035	0.001	0.025	0.000
	-21.04	-32.44	-13.97	-0.513
$T = 100\,sec$	1.288	2.130	0.136	0.152
$dt = 0.01$	0.012	-0.010	0.025	0.000
	-21.02	-32.42	-13.97	-0.513
$T = 10\,sec$	1.130	2.173	0.121	0.182
$dt = 0.01$	0.001	-0.001	0.001	0.000
	-21.03	-32.47	-14.11	-0.543

Table 7.2: Calculated Lyapunov Exponents using Different Parameters

Examination of the values produced by this function indicate that the difference between the two sampling frequencies does not have a very significant affect on the results of the analysis. The values produced indicate that the length of time over which the data spans has a much larger effect. The only major difference between the analyses over one hundred seconds with a sampling time of 0.001 seconds and those with a sampling time of 0.01 seconds is that the time required for the smaller sampling time is an order of magnitude greater. The function output and output plot for each of the sixteen runs can be found in Appendix C1.

When the values for Lorenz-2 system and the Rössler system are compared with the published values, the runs with a sample time of 0.01 seconds for one hundred seconds produce accurate results most efficiently. With each of the three exponents for the Lorenz-2 system differing from the published values by no more than one one-hundredth and the exponents from the Rössler system differing by a maximum of three one-hundredths, it can be concluded that both the function to generate the data sets for the systems and the function to determine the Lyapunov exponents for the systems are functioning properly. Development of the data sets is also validated visually following

its creation. Because of this, the exponents calculated for the Lorenz-1 system and the Duffing system for a sampling time of 0.01 seconds over one hundred seconds will be the basis for comparison of future analyses.

After the discovery that data sets iterated using the smaller sampling times appeared to less efficiently produce near identical results to larger sampling times, the idea was proposed to continue the study utilizing data iterated with larger sampling times. Because all of the other analyses to follow require a pre-existing data set, the next step was to create lengthy data sets for each of the systems using the Lie Series approximations. To further examine how different sampling times affected the results of each analysis, two data sets were created for each system. All eight of the data sets were iterated using a fifth-order Lie Series approximation of their respective systems. Each of the data sets were iterated using the same initial conditions, (1, 0, 0) to create a set of one hundred thousand data points. The following table displays the some of the properties for each data sets created for this study. The display output and time series plots for each of the data sets can be found in Appendix C2.

	Sampling Time	Data Length	Initial Conditions	Added Noise
Lorenz (10,28,8/3)	0.01 sec	1000 sec	(1,0,0)	0.005
Lorenz (10,28,8/3)	0.001 sec	100 sec	(1,0,0)	0.005
Lorenz (16,45.92,4)	0.01 sec	1000 sec	(1,0,0)	0.005
Lorenz (16,45.92,4)	0.001 sec	100 sec	(1,0,0)	0.005
Rössler (.15,.20,10)	0.05 sec	5000 sec	(1,0,0)	0.005
Rössler (.15,.20,10)	0.01 sec	1000 sec	(1,0,0)	0.005
Duffing (.25,.30,1.0)	0.05 sec	5000 sec	(1,0,0)	0.001
Duffing (.25,.30,1.0)	0.01 sec	1000 sec	(1,0,0)	0.001

Table 7.3: Parameters of Simulated Data Sets

The fourth and final column in Table 7.3 lists the magnitude of the random noise that was added to the signal. The noise was added to the signal to limit the number of significant digits and produce more realistic data. Preliminary analyses conducted using noise-free data produced exceedingly large embedding dimension values when employing both the standard method as well as Singular System Approach. Without the additional noise present within the signal, the Singular System Approach suggested that an embedding dimension of more than fifty was necessary to reconstruct the attractor from the classical Lorenz system. While this reconstruction would have accurately presented a recreation of the system's attractor, the precision in the data would far exceed what is necessary for this study. To remedy this situation, the random number generator function in MATLAB was used to add a small amount of noise to the signal and reduce the number of significant digits. The noise added to the system was white noise and was adjusted so that it had a mean value of zero. The magnitude of the noise was controlled by the function that created the data set and is the value presented in Table 7.3. The magnitudes were selected experimentally so that they were as large as possible without producing any visible change in the data plots of each individual variable's time series. Figure 7.1 shows an excerpt of one of the signals produced by the classical Lorenz system and Fig. 7.2 displays the same excerpt with the addition of white noise with a magnitude of 0.005.

By reducing the number of significant digits, the amount of information within the data is reduced and become more comparable to actual data. Because less information exists, fewer dimensions are required when embedding the data.

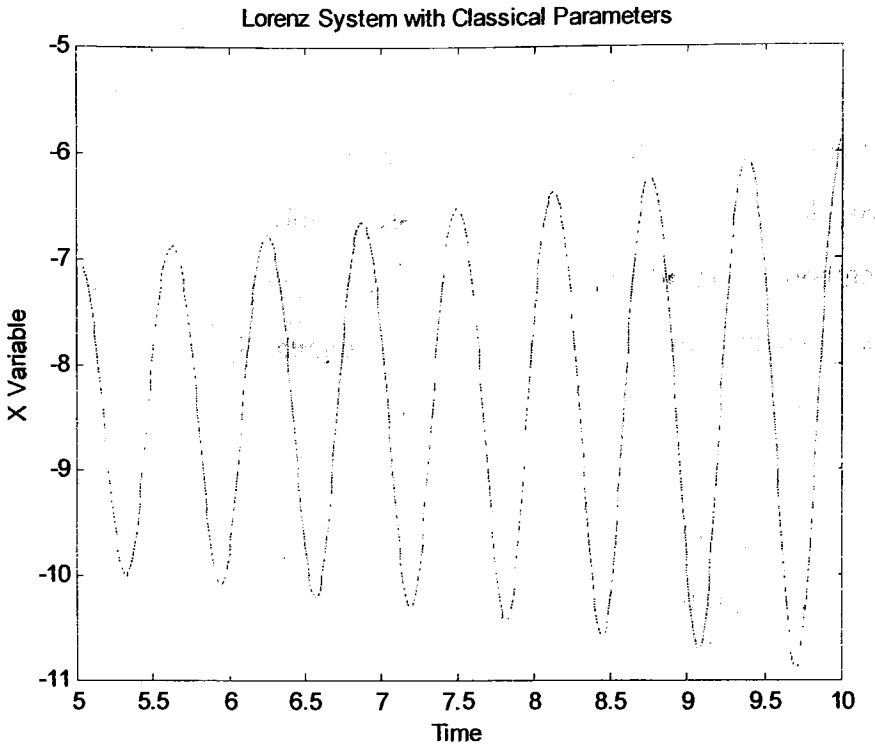


Figure 7.1: Excerpt of Signal from Classical Lorenz System

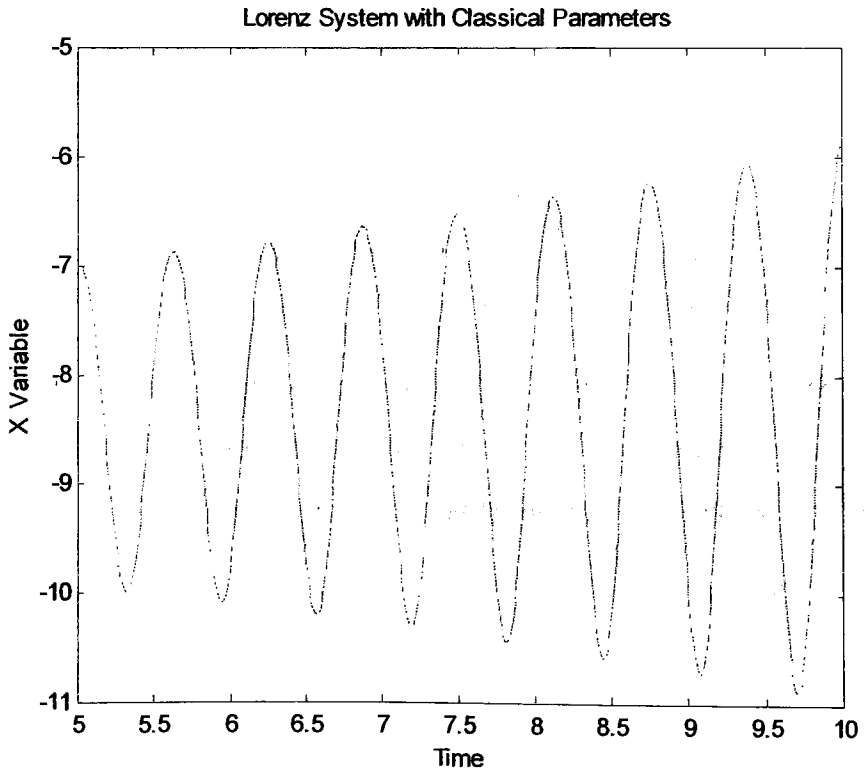


Figure 7.2: Signal from Classical Lorenz System with Added Noise

After further investigation was conducted, it was determined that when working with data from chaotic systems, it is important to be able to examine the longest amount of data from the system with respect to time. In order for this examination to be as efficient as possible, it is important that the sampling time of the data be as large as possible without misrepresenting the structure of the signal. As a result, further analyses done on both of the Lorenz systems were conducted using a sampling time of 0.01 seconds and the Rössler and Duffing systems were examined with sampling times of 0.05 seconds. The other four data sets collected with higher sampling rates were subsequently abandoned in an effort to increase the efficiency of the study.

After calculating the entire spectrum of Lyapunov exponents for each of the systems, the function to determine the largest Lyapunov exponent was applied to each of the systems. Because this function required that many computation cycles, each of which examines a significant amount of data from the data set, the data sets with smaller sampling times were not used. Each of the four systems were examined using ninety-thousand data points. From this data, the function examined three hundred different segments at two-hundred and fifty data points per segment. This number of data points results in two and a half seconds per segment for both of the Lorenz system and twelve and a half seconds per segment for the Rössler system and Duffing system. These parameters are agreeable with each of the systems because the largest Lyapunov exponents for the Rössler system and Duffing system are very close to zero and cause the trajectories to diverge much slower than the trajectories of the Lorenz systems. Table 7.4 presents the largest Lyapunov exponents as calculated by the MATLAB function for the four data sets with larger sampling times. The table also includes the difference between these values and those calculated when the entire Lyapunov spectrum was determined.

	Lorenz (10, 28, 8/3)	Lorenz (16, 45.92, 4)	Rössler (0.15, 0.20, 10)	Duffing (0.25, 0.30, 1.0)
Largest Lyapunov Exponent	1.456	2.176	0.0964	0.0000
Expected	1.287	2.130	0.136	0.152
Difference	0.169	0.046	0.0396	0.152
% Difference	13%	2.2%	29%	---

Table 7.4: Largest Lyapunov Exponents

Of the four systems examined, the largest Lyapunov exponent for the Lorenz system with the parameters $\sigma = 16$, $r = 45.92$, and $b = 4$, and the Rössler system are determined within five one-hundredths of the previous calculation. Because the actual value of the largest Lyapunov exponent for this Lorenz system is much larger than the Rössler system, the percent difference for the Lorenz system is much smaller. The largest Lyapunov exponent for the other Lorenz system, with parameters values of $\sigma = 10$, $r = 28$, and $b = 8/3$, was only determined within seventeen hundredths of the previously calculated value. As the previously calculated value for this system is about an order of magnitude greater than the largest Lyapunov exponent for the Rössler system, the percent difference for this Lorenz system is also smaller.

It would appear that the accuracy of the results from this analysis are dependent upon the value of the largest Lyapunov exponents as well as the other Lyapunov exponents within the spectrum for each system. The second Lorenz system presented in the table above possesses the largest Lyapunov exponent with the greatest magnitude. This system also displays the greatest difference between the largest Lyapunov exponent and the Lyapunov exponent with the largest magnitude in the negative direction. As a result of the great difference between the largest and smallest Lyapunov exponent, the largest exponent influences the trajectory of the system more significantly and the calculations are able to converge much more quickly. Similarly, the Lorenz system presented in the first column holds the largest Lyapunov exponent with the second

greatest magnitude. The difference between the largest Lyapunov exponent of the system and the exponent with the largest magnitude in the negative direction is also the second largest.

Of the four systems examined with this analysis, the results of the system that contained the most discrepancies were from the Duffing system. This could be a result of a number of different factors and probably a combination of many of them. Unlike the other three systems that are being examined, the Duffing system is a non-autonomous system and one of the state variables is time. Because of this, one of the Lyapunov exponents for the system is a zero at all points on the attractor. This factor also produces problems later because the value of the time variable will always be increasing. As a result, the size of the system's strange attractor becomes infinite as time goes to infinity. The other major problem with this system is that unlike the other systems where the difference across all the Lyapunov exponents is 22.3, 34.5, and 14.2, the difference between the largest and smallest Lyapunov exponents for the Duffing system is 0.66. This small difference between the two extreme Lyapunov exponents would indicate that a much greater amount of time and data is required for the analysis to adequately converge to the system's correct largest Lyapunov exponent.

After testing the function to determine the largest Lyapunov exponent from data sets of each of the systems, the fractal dimensions of the four systems were calculated. Three different methods were employed to calculate the fractal dimensions. These three methods were the Correlation Dimension, Capacity Dimension, and Information Dimension. The Correlation Dimension of each of the four systems were calculated using five thousand data points from the previously created data sets. The data with the larger sampling times were used. Two hundred and fifty points were randomly selected from the five thousand points and examined. A Theiler coefficient of ten was used to eliminate spurious structures within the plots of Correlation Sums versus the radius examined.

Because of the inefficient nature of the analyses to determine the Capacity Dimension and the Information Dimension, the number of data points used was reduced to one thousand to allow the calculations to be completed within a reasonable amount of time. Each of the four attractors were then covered with an array of different sized volume elements. Thus the number of inhabited elements and entropy could be calculated to determine the Capacity Dimension and Information Dimension, respectively. The dimensions calculated by each of these analyses for the four systems are presented in Table 7.5.

	Lorenz (10, 28, 8/3)	Lorenz (16, 45.92, 4)	Rössler (0.15, 0.20, 10)	Duffing (0.25, 0.30, 1.0)
Expected	2.062	2.066	2.010	2.296
Correlation	2.050	2.101	1.977	0.7437
Capacity	1.653	1.674	1.584	1.119
Information	1.674	1.750	1.798	1.122

Table 7.5: Fractal Dimensions of Four Chaotic Systems

After calculating the three dimensions, the Correlation Dimension was selected to be used further in the study while the other two were abandoned. The fractal dimensions calculated for the Duffing equation proved to show the most deviation from the expected values. It is thought that this deviation is caused by the purely linear state variable of time and the infinite size of the system's attractor. It is hypothesized that more accurate values could be produced by increasing the size of the data subset used for the different fractal dimension calculations. Because such a long amount of time would be required to perform the calculations with substantially larger data sets, the Duffing equation will no longer be included in the study.

The fractal dimension values of the Correlation Dimension for the three remaining systems were found to be within 0.04 of the anticipated values. As each of the three systems are three-dimensional chaotic systems, the fractal dimensions should have values between two and three. Only one of the three Correlation Dimension values were calculated to be less than two but it was still within 0.04 of the value anticipated. The

difference between the Capacity Dimensions and Information Dimensions calculated and those anticipated were of an order of magnitude greater than the differences in the Correlation Dimension values. All of the Capacity Dimension and Information Dimension values were substantially less than two and the Correlation Dimension was selected to calculate fractal dimensions for the remainder of the study.

7.3 RECONSTRUCTION ANALYSIS

After selecting and testing the analysis methods that will be used to examine the chaotic attractors, data was taken from each of the systems to be used to reconstruct the system's strange attractor. The x variable was selected from each of the Lorenz systems and the Rössler system. The main method of attractor reconstruction used was the Delay Vector Space Method. This was done using the delay time value from the first zero of the autocorrelation, the first minimum of the autocorrelation, and the first inflection point of the autocorrelation. The delay time values for half the maximum value, one-e^{th} of the maximum value, and one-tenth of the maximum value were not used because they were determined experimentally for specific attractors. Delay time values were also acquired using the Average Displacement Method and the Average Mutual Information Methods. The final method to reconstruct the strange attractors employed the Singular System Approach. Table 7.6 shows the delay time values calculated for each of the different methods using the Delay Vector Space Method.

	Lorenz-1	Lorenz-2	Rössler
Autocorrelation, First Zero	1.15	4.38	1.50
Autocorrelation, First Local Minimum	0.80	0.75	3.00
Autocorrelation, First Inflection Point	0.16	0.10	1.45
Average Displacement Method	0.07	0.05	0.45
Average Mutual Information Method	0.21	0.14	0.90

Table 7.6: Delay Times for Attractor Reconstruction in Seconds

Each of the values presented in Table 7.6 were calculated using the methods described in Chapter Three, implemented into the software described in Chapter Six. The

output from each of the programs can be found in Appendices C7, C8 and C9. The appendices also include information from the reconstruction using the Singular System Approach. Initial examination of the delay time values and a two-dimensional projection of the reconstructed attractors show that both the first zero and the first local minimum of the autocorrelation function appear to be too large. The two-dimensional projections for each of the reconstructions using these delay time values contained a great deal of irrelavance. The other three methods appeared to produce smaller values with the Average Displacement Method producing the smallest value for each system.

Following the calculation of the various delay time values, it was necessary to determine the embedding dimension to be used. Each of the three systems were examined to determine what embedding dimension was necessary to adequately distribute the data throughout the embedding space. The attractor for each of the three systems was reconstructed using the delay time value of the first inflection point of the autocorrelation, with embedding dimensions ranging from three to ten. The embedding dimensions that proved to be large enough to sufficiently distribute the data are presented in Table 7.7.

	Lorenz-1	Lorenz-2	Rössler
Embedding Dimension	7	7	8

Table 7.7: Embedding Dimensions Required for Attractor Reconstruction

With the necessary embedding dimensions determined, the attractor for each of the systems could be reconstructed. The next step in determining the optimal reconstruction method was to determine which reconstruction maintained the original system's fractal dimension. Since the Correlation Dimension was chosen to determine the fractal dimensions for the three systems, it was applied to each of the reconstructed attractors for all three of the systems. These calculations were done using the same parameters as the original calculation of the Correlation Dimensions for the three original

systems. Table 7.8 shows the Correlation Dimensions calculated from each of the different reconstructions.

	Lorenz-1	Lorenz-2	Rössler
Expected	2.050	2.101	1.977
Autocorrelation, First Zero	3.85	5.06	2.16
Autocorrelation, First Local Minimum	3.59	4.89	2.45
Autocorrelation, First Inflection Point	2.08	2.09	2.16
Average Displacement Method	1.85	1.96	1.96
Average Mutual Information Method	2.08	2.08	2.07
Singular System Approach	1.55	1.71	1.89

Table 7.8: Correlation Dimensions of Reconstructed Attractors

As predicted from the two-dimensional projections, the attractors reconstructed using the first zero and first local minimum of the autocorrelation produced Correlation Dimensions that differ the most from the values previously calculated. In the case of the two Lorenz systems, the Correlation Dimensions using these two methods were calculated to be greater than three. The Correlation Dimensions calculated for the attractors reconstructed using the Singular System Approach were the lowest of the examined methods. Each of these three Correlation Dimensions are less then two. The Correlation Dimensions for the attractors reconstructed using the Average Displacement Method are also lower than two. The reconstruction methods that performed the best were the Delay Vector Space Method using the first inflection point of the autocorrelation function and the first local minimum of the Average Mutual Information method. The reconstructions using the first inflection point produced Correlation Dimensions with a total difference from the expected values of less than ten percent between the three systems. The Average Mutual Information Method produced slightly better Correlation Dimensions with a total difference of less than five percent between the three systems.

The next step to determine the optimal method for attractor reconstruction was to examine the exponential divergence of nearby trajectories within the reconstructed attractor and determine which method best maintained the largest Lyapunov exponent of

the original system. The calculation was also done using the same parameter values as when the original systems were examined. The only change was that the Euclidean distance was calculated for seven- and eight-dimensional systems as opposed to the original three-dimensional systems. Table 7.9 shows the values for the largest Lyapunov exponents of the various reconstruction methods for the three systems.

	Lorenz-1	Lorenz-2	Rössler
Expected Value	1.456	2.176	0.0964
Autocorrelation, First Zero	0.909	0.945	0.090
Autocorrelation, First Local Minimum	1.096	1.158	0.105
Autocorrelation, First Inflection Point	1.290	2.156	0.088
Average Displacement Method	1.544	2.281	0.100
Average Mutual Information Method	1.256	2.069	0.090
Singular System Approach	1.757	2.445	0.121

Table 7.9: Largest Lyapunov Exponent Calculated from Reconstructed Attractors

The values in the table above show once again that the attractors reconstructed using the delay time values of the first zero and first local minimum of the autocorrelation function have the least in common with the values of the original systems. Of the remaining four, the attractors reconstructed using the first inflection point of the autocorrelation function and the first local minimum of the average mutual information function display exponential divergence that most resembles the original systems. When examining the largest Lyapunov exponents of the reconstructed attractors, the reconstruction method using the delay time value of the first inflection point of the autocorrelation function shows the best results with a total difference of less than 0.07 from the expected values across the three systems. The second best reconstruction produced largest Lyapunov exponents with a total difference of less than 0.15 from the expected values across the three systems using the Average Mutual Information method.

8 CONCLUSION

8.1 GENERAL

From the results of this study presented in Chapter Seven, the two best methods for attractor reconstruction employing the Delay Vector Space method with the delay time value of the first inflection point of the autocorrelation and the delay time value of the first local minimum of the average mutual information function. The study shows that the reconstruction using the delay time value of the first inflection point of the autocorrelation function produced the best recreation of the exponential divergence of nearby trajectories. The study also shows that the fractal structure maintained by the attractor reconstructed using the Average Mutual Information Method surpassed the other reconstruction methods. Since the results produced by these two methods are so close, it is not possible to determine which of the two is the superior method in the time allotted for this study. However, in the event that the precision required in an analysis of a chaotic time series is less stringent than the performance of each of the methods, the choice of methods falls onto the most efficient one. Because of the ease in which the autocorrelation of a signal can be calculated, the delay time of the first inflection point of the autocorrelation function can be calculated in less time than the first local minimum of the average mutual information function. As a result, the reconstruction method using the Delay Vector Space Method and the time delay value of the first inflection point of the autocorrelation function produces the optimal attractor reconstruction for the precision level maintained within the study.

8.2 APPLICATION OF ACTUAL SYSTEMS

After determining which of the reconstruction methods showed the highest level of performance, the reconstruction method as well as analyses of the fractal structure and exponential divergence of nearby trajectories were applied to data collected from the two

actual nonlinear systems described in Chapter Five, Section Three. Data was collected from one of the variables for each of the systems. The autocorrelations of this data were then calculated and the first inflection points were located. Using the associated delay time values, each data set was embedded with a range of embedding dimensions, after which the optimal embedding dimension was chosen. Once the data set was used to reconstruct each of the system's strange attractors, the Correlation Dimensions and largest Lyapunov exponents were calculated. The details of the reconstruction and analysis of each of these systems is presented in Table 8.1.

	Double-Well Oscillator	Chua's Circuit
Time Step	0.001 seconds	0.00001 seconds
Number of Data Points	60,000	10,000
Data Length	60 seconds	0.1 seconds
Delay Time, First Inflection Pt	0.029 seconds	0.00010 seconds
Embedding Dimension	7	9
Correlation Dimension	2.238	1.944
Largest Lyapunov Exponent	20.05	1558.9

Table 8.1: Detail of Reconstruction and Analysis of Collected Data

All of the output data and output figures from each of the analyses run on these two systems can be found in Appendices C13 and C14. From these results, it can be seen that the delay time value of the first inflection point of the autocorrelation function is smaller than the delay time values obtained from other locations on the autocorrelation function. As the basic model for each of the systems include three state variables, the embedding dimension required to reconstruct each of the system's attractors is consistent with Takens statement that the embedding dimension must be greater than or equal to twice the dimension of the system plus one.

The Correlation Dimension of the data set collected from the Double-Well Oscillator indicates that the system is much more chaotic than any of the systems simulated. This additional amount of chaos could be caused by the particular set of parameter values of the system or a result of the combination of the nonlinear material

properties of the components of the Double-Well Oscillator and the nonlinear dynamics. The largest Lyapunov exponent of the data collected from this system is also much higher than those previously observed. This is a result of the higher level of chaos in this system as well as the very high frequency at which the system is operating. Because the Lyapunov exponents are inversely proportional to the time over which the trajectories diverge, very small amounts of time result in large Lyapunov exponents. In the case of the largest Lyapunov exponents, comparisons are most useful when they are obtained from a common system with different control parameters.

The results of the analysis of the data collected from the Chua's Circuit show a much lower Correlation Dimension. It would appear that the data set is so weakly chaotic that the analysis produced a Correlation Dimension slightly smaller than two. From the precision seen previously with these calculations, this would indicate that the actual Correlation Dimension is not much more than 2.01. The second analysis of this data set shows that the largest Lyapunov exponent is equal to 1508.9. As with the other physical system, the magnitude of this value is due to the extremely high frequency at which the system is oscillating.

8.3 RECOMMENDATIONS FOR FUTURE WORK

Although the majority of this study was conducted on what was a high-performance personal computer during the time of the study, more advanced systems will become available due to the exponential growth of technology. With the aid of a more powerful system, a number of different steps could be taken to continue this study. Using faster hardware, much larger data sets could be analyzed in the same amount of time. With more data from each system for the analysis, it would be possible to increase the precision in the analyses. More advanced hardware could also be used to perform a much larger number of iterations in the same amount of time required during this study.

An increased number of iterations would allow the results to converge with higher precision.

The ability to examine larger data sets with more advanced hardware could enable this study to be successfully performed on data produced by the Duffing equation. The larger data set would enable the algorithms to converge to more accurate values for systems containing purely linear variables. With the ability to perform quicker, more efficient calculations, it should be possible to examine the fractal structure of even a system with an infinite volume strange attractor. Once the analysis has been successfully adapted to apply to data produced by the Duffing equation, it could be applied to other nonautonomous, low-dimensional chaotic systems.

In addition to including other nonautonomous systems in this study, the source of the data sets can also be broadened to include other autonomous systems. This study was limited to two variations of the Lorenz system and one of the Rössler system but would greatly benefit from the application of these methods to other systems. Application of this procedure to multiple low-dimensional chaotic signals could further support its successful ability to determine the optimal reconstruction method for reconstructing a strange attractor from the time series of a single chaotic variable. The range of systems to which this process is applied could also be extended to include signals produced by higher-order chaotic systems. As this is done, the necessary modification and adjustments could be made to enable these methods to choose the best attractor reconstruction method.

Further work can be done by including data sets from quasi-periodic systems and stochastic systems. A quasi-peroidic signal is a signal created by the superposition of multiple harmonic signals where the ratio of the frequencies is an irrational number. Results from these systems can then be used to verify that the analyses are able to identify the chaotic behavior and distinguish it from behavior that is not chaotic. While the data produced by quasi-periodic and stochastic systems may appear to be chaotic, it

does not contain the intrinsic structure present within chaotic signals produced by nonlinear systems.

In addition to increasing the number of systems used to produce the data sets, further work could be done by increasing the number of attractors reconstructed with each method. This will provide a better understanding of just how well the structure of the original attractor is preserved when the attractor is reconstructed. The additional data sets can be acquired by iterating each of the systems across a range of initial conditions. Because the data would be acquired from the same system, the dynamic properties of the different data sets will remain the same, but the different initial conditions will provide enough variation to enable the study to determine the precision of the results.

Increasing the size of the data sets, the number of data sets, and broadening the range of element sizes will enable the Capacity Dimension and Information Dimension of a chaotic system to be more successfully utilized. Using more powerful hardware, it would be possible to cover a reconstructed attractor with a seven- or eight-dimensional array of volume elements. By studying the relationship between the distribution of the attractor within the array and the size of the volume elements, it would be possible to determine Capacity and Information Dimensions. This would provide additional means to determine the fractal dimension of a reconstructed attractor and a way to verify the values produced by the algorithm used to calculate the Correlation Dimension.

In addition to studying how the fractal structure and exponential divergence of nearby trajectories are maintained in a reconstructed attractor, other properties of chaotic signals can be examined. This could include some of the newer algorithms including calculating the d_∞ parameter[4] or the attractors anisotropicity[25]. The relationship between the delay time value and the embedding dimension could also be explored further. Because the embedding dimension is a function of the delay time value to some extent, it may be possible to improve the results of reconstruction using other delay values if they are used to calculate the embedding dimension on an attractor-by-attractor

basis. Finally, this study can be applied to other methods for attractor reconstruction as they are developed. This could include methods such as using the C-C Method[17] to calculate the delay time value for reconstruction in Delay Vector Space.

RESOURCES

1. Abarbanel, H.D.I., Frison, T.W. Tsimring, L.S., Obtaining Order in a World of Chaos Time Domain Analysis of Nonlinear and Chaotic Signals, IEEE Signal Processing Magazine, May 1998.
2. Baker, G.L., Gollub, J.P., Chaotic Dynamics: An Introduction, Cambridge University Press: New York, 1990.
3. Beckers, F., Ramaekers, D., Aubert, AE., Nonlinear Dynamics in Heart Rate Variability, Computers in Cardiology, Vol. 27, pp. 131-134, 2000.
4. Bonasera, A., Bucolo, M., Fortuna, L., Rizzo, A., The d_∞ Parameter to Characterise Chaotic Dynamics., Proceedings of the International Joint Conference on Neural Networks, vol. 5, pp. 565-570, 2000.
5. Broomhead, D. S., King, G. P., Extracting Qualitative Dynamics from Experimental Data, Physica D, Vol. 20, pp. 217-236, 1986.
6. Chen, B., Wang, N., Determining EMG Embedding and Fractal Dimensions and its Application, 22nd Annual EMBS International Conference, Chicago, IL., July 23-28, 2000.
7. Christensen, S.R., Zohdy, M.A., Analysis of Chaotic Physical Systems and an Algorithm for Control, Proceedings of the American Control Conference, Philadelphia, PA, June 1998.
8. Close, Charles M., Frederick, Dean K., Newell, Jonathan C., Modeling and Analysis of Dynamics Systems, John Wiley & Sons, Inc.: New York, 2002.
9. Fraser, A. M., Swinney, H. L., Independent Coordinates for Strange Attractors from Mutual Information, Physical Review A, Vol. 33, No. 2, pp. 1134-1140, February 1986.
10. Grassberger, P. in: Holden, Arun V. (Ed.), Estimating the Fractal Dimension and Entropies of Strange Attractors, Chaos, Princeton University Press: Princeton, New Jersey, pp. 291-311, 1986.
11. Grassberger, P., Procaccia, I., Characterization of Strange Attractors, Physical Review Letters, Vol. 50: pp. 346-349, 1983.
12. Grassberger, P., Procaccia, I., Measuring the Strangeness of Strange Attractors, Physica D, Vol. 9: pp. 189-208, 1983.
13. Han, M., Xi, J., Chaotic System Identification Based on Kalman Filter, Proceedings of the International Joint Conference on Neural Networks, Vol. 1, pp. 675-680, 2002.
14. Hilborn, Robert C. Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers, Oxford University Press: New York, 1994.
15. Horneo, R. et al., Nonlinear Analysis of Time Series Generated by Schizophrenic Patients, Engineering in Medicine and Biology, IEEE Engineering in Medicine and Biology, Vol. 18, No. 3, pp. 84-90, May-June 1999.
16. Kaplan, J. L., Yorke, J. A., Chaotic Behavior of Multidimensional Difference Equations, Lecture Notes in Mathematics, Vol. 730, pp. 204-227, 1979.

17. Kim, H. S., Eykholt, R., Salas, J. D., Nonlinear Dynamics, Delay Times, and Embedding Windows, Physica D, Vol. 127, pp. 48-60, 1999.
18. King, G. P., Jones, R., Broomhead, D. S., Phase Portraits from a Time Series: A Singular System Approach, Nuclear Physics B, Vol. 2, pp. 379-390, 1987.
19. Kocak, K., Saylan, L., Sen, O., Nonlinear Time Series Prediction of O_3 Concentration in Istanbul, Atmospheric Environment, Vol. 34, pp. 1267-1271, 2000.
20. Krot, A. M., Chaotic Dynamic Methods Based on Decomposition of Vector Functions in Vector-Matrix Series into State-Space, 10th Mediterranean Electrotechnical Conference, MEleCon, Vol. II, 2000.
21. Krot, A.M., Minervina, H.B., Biomedical Signal Processing Based on Estimation of Minimal Dimensional Embedding of Chaotic Attractors, Proceedings of the Mediterranean Electrotechnical Conference, MEleCon, Vol. 2, pp. 733-736, 2000.
22. Lei, M., Wang, Z., Fen, Z., A Method of Embedding Dimension Estimation Based on Symplectic Geometry, Physical Letters A, Vol. 303, pp. 179-189, 2002.
23. Lorenz, E. N., Deterministic Nonperiodic Flow, Journal of the Atmospheric Sciences, Vol. 20, pp. 130-141, March 1963.
24. Mathew, Manu K., Nonlinear System Identification and Prediction, Master's Thesis, RIT, 1993.
25. Michieli, I., Vojnovic, B., On Reconstruction of Strange Attractors Using their Noisere Related Directional Properties, Signal Processing, Vol. 82, pp.1443-1453, 2002.
26. Miyoshi, T., Nitanai, T., Inaba, N, Chaotic Attractor with a Characteristic of Torus, IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 47, No. 6, June 2000.
27. Moon, Francis C. Chaotic and Fractal Dynamics: An Introduction for Applied Scientists and Engineers, John Wiley & Sons, Inc.: New York, 1992.
28. Nichols, J. M., Nichols, J. D., Attractor Reconstruction for Non-linear Systems: A Methodological Note, Mathematical Biosciences, Vol. 171, pp. 21-32, 2001.
29. Otani, M, Jones, A. J., Automated Embedding and the Creep Phenomenon in Chaotic Time Series, Unpublished, 14 October, 2000.
30. Packard, N. H., Crutchfield, J. P., Farmer, J. D., Shaw, H. S., Geometry from a Time Series, Physical Review Letters, Vol. 45, pp. 712-716, 1980.
31. Parker, T.S., Chua, L.O. Practical Numerical Algorithms for Chaotic Systems, Springer-Verlag: New York, 1989.
32. Petry, A., Barone, D.A.C., Fractal Dimension Applied to Speaker Identification, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, Vol. 1, pp. 405-408, 2001.
33. Proakis, John G., Manolakis, Dimitris G., Digital Signal Processing: Principles, Algorithms, and Applications, Prentice Hall: Upper Saddle River, New Jersey, 1996.
34. Radojicic, I., Mandic, D., Vulic, D., On the Presence of Deterministic Chaos in HRV Signals, Computers in Cardiology, Vol. 28, pp. 465-468, 2001.
35. Ramchurn, S.K., Baijnath, D., Murray, A., Low-dimensional Chaotic Behavior in Heart-Rate Variability, Computers in Cardiology, Vol. 27, pp. 473-476, 2000.

36. Ripoli, A., Emdin, M., Complexity of Heart Rate, Blood Pressure and Respiration Disclosed by Pattern Fractal Analysis, Computers in Cardiology, Vol. 27, pp. 135-138, 2000.
37. Robertson, William., Analysis of Deterministic Chaotic Signals, Master's Thesis, RIT, 1994.
38. Rosenstein, M. T., Collins, J. J., De Luca, C. J., Reconstruction Expansion as a Geometry-Based Framework for Choosing Proper Delay Times, Physica D, Vol. 73, No. 1-2, pp. 82-98, 15 May, 1994.
39. Rössler, O.E. in: Holden, Arun V. (Ed.), How Chaotic is the Universe?, Chaos, Princeton University Press: Princeton, New Jersey, pp. 315-320, 1986.
40. Smario, David J., Multicorrelation Analysis and State Space Reconstruction, Master's Thesis, RIT, 1994.
41. Strogatz, Steven H. Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering, Perseus Publishing: Cambridge, Massechusetts, 1994.
42. Takens, F. in: Rand, D. Young, L. S. (Eds.), Detecting Strange Attractors in Turbulence, Dynamical Systems and Turbulence, Lecture Notes in Mathematics 898, Berlin: Springer-Verlag, pp. 366-381, 1981.
43. Theiler, J., Spurious Dimension from Correlation Algorithms Applied to Limited Time-Series Data, Physical Review A, Vol. 34: pp. 2427-2432, 1986.
44. Török, J.S., A Constructive Method for Locating Periodic Solutions of Linear and Nonlinear Systems, DE - vol. 56, Dynamics and Vibration of Time-varying Systems and Structures, ASME, 1993.
45. Török, J.S., Estimation of Lyapunov Exponents using a Semi-Discrete Formulation, Applied Mechanics Reviews, Vol. 46, No. 11, 1993.
46. Török, J.S., Symbolic Computation of Lie Series for the Solution of Ordinary Differential Equations, ASEE Annual Conference Proceedings, Session 3565, 1991.
47. Török, J.S., Advani, S.H., Continuous Transformation Groups and Series Solution of Initial Value Problems, International Journal of Nonlinear Mechanics, Vol. 20, No. 4, 1985.
48. Tsonis, Anastasios A., Chaos: From Theory to Application, Plenum Press: New York, 1992.
49. Vallverdu, M. et al., Heart Rate Variability Characterization: Time-Frequency Representation and Nonlinear Analysis, Computers in Cardiology, Vol. 26, pp. 257-260, 1999.
50. Vasilik, A., Numerical Solutions of Ordinary Differential Equations Using Lie Series, Project with paper, RIT, 2002.
51. Wolf, A., in: Holden, Arun V. (Ed.), Qualifying Chaos with Lyapunov Exponents, Chaos, Princeton University Press: Princeton, New Jersey, pp. 273-290, 1986.
52. Wolf, A., Swift, J.B., Swinney, H.L., Vastano, J.A., Determining Lyapunov Exponents from a Time Series, Physica D, Vol. 16: pp. 285-317, 1985.
53. Yang, T., Li, X. F., Shao, H. H., Chaotic Synchronization Using Backstepping Method with Application to the Chua's Circuit and Lorenz System, Proceedings of the American Control Conference, Arlington, VA June 25-27, 2001.

54. Yang, Z.R., Zwolinski, M., Mutual Information Theory for Adaptive Mixture Models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No. 4, April 2001.
55. Zhang, J., Man, K.F., Ke, J.Y., Time Series Prediction using Lyapunov Exponents in Embedding Phase Space, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, vol. 2, pp. 1744-1749, 1998.
56. Ziegler, Edward H. IV., Nonlinear System Identification, Master's Thesis, RIT, 1994.

APPENDICES

The Maple program presented in Appendix A shows how to determine the Lie Series Approximation for the Lorenz system. Similar programs were developed for the other systems. Programs similar to the one presented in Appendix B7 were also created to determine the Jacobian for the other systems. There were additional sections within the program presented in Appendix B9 for the other systems included within the study.

APPENDIX A

```
> restart;
> # This code computes the Nth-order series approximation of the solution to a system of
3 first-order differential equations. The time advance variable is denoted by h.
> #
> with(linalg):
Warning, the protected names norm and trace have been redefined and unprotected

> # Define the right-hand side of the equation.
> F1:=sigma*(x2-x1):
> F2:=x1*r-x2-x1*x3:
> F3:=x1*x2-b*x3:
> # Number of terms
> N:=3:
> vars:=[x1,x2,x3]:
> vec:=array([F1,F2,F3]):
> w:=array([x1,x2,x3]):
>     xh:=x1:
>     yh:=x2:
>     zh:=x3:
> for k from 1 to N do
>         J:=jacobian(w,vars):
>         Uw:=evalm(J&*vec):
>     xh:=xh+Uw[1]*(h^k)/factorial(k):
>     yh:=yh+Uw[2]*(h^k)/factorial(k):
>     zh:=zh+Uw[3]*(h^k)/factorial(k):
>    w:=Uw:
> od:
> # Print the Nth order series approximation
> xh:
> yh:
> zh:
> evalm(J):
```

APPENDIX B1

```
%  AUTOCORRELATION_DELAY.M
%
%  This function will calculate the autocorrelation of the input signal
%  and examine the first zero location, the location of the first local
%  minimum, the location where the autocorrelation is half of the maximum,
%  the location where the autocorrelation is (1/e)th of the maximum, the
%  location where the autocorrelation is equal to one-tenth of the maximum,
%  and where the first inflection point occurs on the autocorrelation of
%  the signal.  These delay values will then be used to reconstruct the
%  strange attractor of the system from which the data was collected.
%
%  XYZ->        time series from mat file
%  dt->         sampling time from mat file
%  Trans_Time-> transient time removed from signal(seconds)
%  Data_Length-> number of data points to be examined
%
%
[zero,minimum,half,eth,tenth,inflection]=autocorrelation_delay(XYZ,dt,Trans_Time,Dat
a_Length)
%
%  Copyright (c) 2002-2003 Andrew Dick

function
[zero,minimum,half,eth,tenth,inflection]=autocorrelation_delay(XYZ,dt,Trans_Time,Dat
a_Length);

%  Display Program Name, Date, and Time Information
%  ************************************************
tic;
disp(['*****************************************************************'
]);
disp(['Delay Time value calculation program using autocorrelation by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp([' ']);

%  Preprocess Data
%  ************************************************
LengthOfData=length(XYZ);
DataStart=Trans_Time/dt+1;
if Data_Length+DataStart>LengthOfData;
  DataEnd=LengthOfData;
else;
  DataEnd=Data_Length+DataStart;
```

```
end;
x=XYZ(DataStart:DataEnd,1);
L=length(x);

disp(['Data previously iterated using 5th order Lie Series approximation']);
disp(['Data set consisted of ',num2str(L),' data points with a time step of ',num2str(dt),'
seconds']);
disp(['A transient period of ',num2str((DataStart-1)*dt),' seconds was removed']);
disp([' ']);



%  Calculate and Plot Autocorrelation of Data
%  ************************************************
autoc=xcorr(x,x);
delayt=[-(L-1):1:(L-1)];
figure;
plot(delayt,autoc);
title(['Autocorrelation of Signal']);
xlabel(['Delay Time Steps, dt=',num2str(dt)]);
ylabel(['Magnitude of Autocorrelation']);

acmax=max(autoc);

%  Determine Delay Value for First Zero
%       Add to Plot of Autocorrelation
%  ************************************************
i=1;

acval=acmax;
while acval>0;
   delay=L-1+i;
   acval=autoc(delay);
   i=i+1;
end;
hold on;
plot(delayt(delay),autoc(delay),'.k');
firstzerodelay=delay-L;
disp(['Delay Value at First Zero is ',num2str(firstzerodelay*dt),' seconds']);

%  Determine Delay Value for First Local Minimum
%       Add to Plot of Autocorrelation
%  ************************************************
i=1;
diff=-1;
while diff<0;
```

```
   delay=L-1+i;
   diff=autoc(delay+1)-autoc(delay);
   i=i+1;
end;
plot(delayt(delay),autoc(delay),'.k');
firstlocalminimumdelay=delay-L;
disp(['Delay Value at First Local Minimum is ',num2str(firstlocalminimumdelay*dt),'
seconds']);

%  Determine Delay Value for Half Maximum Value
%        Add to Plot of Autocorrelation
%  ************************************************
i=1;
diff=1;
while diff>0;
   delay=L-1+i;
   diff=autoc(delay)-(acmax/2);
   i=i+1;
end;
plot(delayt(delay),autoc(delay),'.k');
halfmaximumdelay=delay-L;
disp(['Delay Value at Half of Maximum Value is ',num2str(halfmaximumdelay*dt),'
seconds']);

%  Determine Delay Value for 1/e of Maximum Value
%        Add to Plot of Autocorrelation
%  ************************************************
e=exp(1);
i=1;
diff=1;
while diff>0
  delay=L-1+i;
  diff=autoc(delay)-(acmax/e);
  i=i+1;
end;
plot(delayt(delay),autoc(delay),'.k');
enegonemaximumdelay=delay-L;
disp(['Delay Value at 1/e of Maximum Value is ',num2str(enegonemaximumdelay*dt),'
seconds']);

%  Determine Delay Value for One-Tenth Maximum Value
%        Add to Plot of Autocorrelation
%  ************************************************
i=1;
diff=1;
while diff>0;
```

```
  delay=L-1+i;
  diff=autoc(delay)-(acmax/10);
  i=i+1;
end;
plot(delayt(delay),autoc(delay),'.k');
onetenthmaximumdelay=delay-L;
disp(['Delay Value at One-Tenth of Maximum Value is
',num2str(onetenthmaximumdelay*dt),' seconds']);

%  Determine Delay Value for First Inflection Point
%       Add to Plot of Autocorrelation
%  ************************************************
dautoc=zeros(2*L-1,1);
for i=2:(2*L-2)
   dautoc(i)=(autoc(i-1)-autoc(i+1))/(2*dt);
end;
i=1;
diff=1;
while diff>0;
   delay=L-1+i;
   diff=dautoc(delay+1)-dautoc(delay);
   i=i+1;
end;
plot(delayt(delay),autoc(delay),'.k');
firstinflectionpointdelay=delay-L;
hold off;
disp(['Delay Value at First Inflection Point is ',num2str(firstinflectionpointdelay*dt),'
seconds']);

%  Plot Derivative of Autocorrelation of Signal
%  ************************************************
figure;
plot(delayt,dautoc);
title(['Derivative of Autocorrelation of Signal']);
xlabel(['Delay Time Steps, dt=',num2str(dt)]);
ylabel(['Magnitude of Derivative of Autocorrelation']);
hold on;
plot(delayt(delay),dautoc(delay),'.k');
hold off;

%  Reconstruct Attractor with Each of the Six Values
%  ************************************************
d(1)=firstzerodelay;zero=d(1)*dt;
d(2)=firstlocalminimumdelay;minimum=d(2)*dt;
d(3)=halfmaximumdelay;half=d(3)*dt;
d(4)=enegonemaximumdelay;eth=d(4)*dt;
```

```
d(5)=onetenthmaximumdelay;tenth=d(5)*dt;
d(6)=firstinflectionpointdelay;inflection=d(6)*dt;
Ld=length(d);
for i=1:Ld;
  clear x1;clear x2;
  for j=1:L-d(i);
    x1(j,1)=x(j+d(i));
    x2(j,1)=x(j);
  end;
  figure;
  plot(x1,x2);
  title(['Time Delay of ',num2str(d(i)*dt),' seconds']);
  xlabel('Data from time series');
  ylabel('Delayed data from time series');
end;

disp([' ']);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-(minutes*60);
disp(['Elapsed time ',num2str(minutes),' minutes, ',num2str(seconds),' seconds']);
disp('Normal Termination of Program AUTOCORRELATION_DELAY.M');
disp(['**************************************************************'
]);
```

APPENDIX B2

```
%  AVERAGE_DISPLACEMENT.M
%
%  This function will calculate a delay value to be used to
%  reconstruct the strange attractor of the system from which
%  the data was collected.  This value will be determined by
%  calculating the average displacement of the recreated vectors
%  and determining where the average displacement curve has a
%  slope that is less than forty percent of the slope at a
%  delay value of one.
%
%  XYZ->        time series from mat file
%  dt->         sampling time from mat file
%  m->          embedding dimension
%  taus->       maximum delay value to examine
%  trans_time-> length of transience to remove (seconds)
%  data_length-> number of data points to be examined
%
%  [ave_disp_min]=average_displacement(XYZ,dt,m,taus,trans_time,data_length)
%
%  Copyright (c) 2002-2003 Andrew Dick

function [ave_disp_min]=average_displacement(XYZ,dt,m,taus,trans_time,data_length);

%  Display Program Name, Date, and Time Information
%  ************************************************
tic;
disp(['*************************************************************************'
]);
disp(['Delay Time value calculation program using average displacement by Andrew
Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp([' ']);

%  Preprocess Data
%  ***********************************************
trans=trans_time/dt+1;
L1=length(XYZ);
if L1<trans-1+data_length
  data_end=L1;
else;
  data_end=trans-1+data_length;
end;
```

```
X=XYZ(trans:data_end,1);
L=length(X);

disp(['Data previously iterated using 5th order Lie Series approximation']);
disp(['Data set consisted of ',num2str(L),' data points with a time step of ',num2str(dt),'
seconds']);
disp(['A transient period of ',num2str((trans-1)*dt),' seconds was removed']);
disp([' ']);

%  Nested Summations
%  ******************************************************
S=zeros(taus+1,1);
for tau=1:(taus+1);
  N=L-(m-1)*(tau-1);
  rootsum=0;
  for i=1:N;
    sum=0;
    for j=1:m;
      sum=sum+(X(i+(j-1)*(tau-1))-X(i))^2;
    end;
    rootsum=rootsum+sqrt(sum);
  end;
  S(tau)=S(tau)+rootsum/N;
  delay(tau)=(tau-1)*dt;
end;

%  Plot of Average Displacement
%  ******************************************************
figure;
plot(delay,S)
title('Average Displacement');
xlabel('Delay Time, sec');
ylabel('Average Displacement Value');

%  Calculation and Plot of First Derivative
%  ******************************************************
dS=zeros(taus-2,1);
for Si=2:taus-1;
  dS(Si-1)=(S(Si+1)-S(Si-1))/(2*dt);
  ddelay(Si-1)=(Si-1)*dt;
end;

%  Locate Delay Value where Slope is 40% of slope at S(1)
%       Add to plot of First Derivative of Average Displacement
%  ******************************************************
index=0;
```

```
diff=1;
while diff>0;
  index=index+1;
  diff=dS(index)-0.4*dS(1);
end;
avedispmethod=index;ave_disp_min=avedispmethod*dt;

disp(['Delay Value of ',num2str(avedispmethod*dt),' seconds at 40% of slope at S(1)'])

hold on;
plot(delay(index+1),S(index+1),'.k');
hold off;
figure;
plot(ddelay,dS);
title('Slope of Average Displacement');
xlabel('Delay Time, sec');
ylabel('Slope of Average Displacement Value');
hold on;
plot(ddelay(index),dS(index),'.k');
hold off;


%  Reconstruction of Attractor Using Delay Time Value
%  ****************************************************
for i=1:L-avedispmethod;
  x1(i)=X(i+avedispmethod);
  x2(i)=X(i);
end;

%  Plot Reconstructed Attractor
%  ****************************************************
figure;
plot(x1,x2);
title(['Time Delay of ',num2str(avedispmethod*dt),' seconds']);
xlabel('Data from time series');
ylabel('Delayed data from time series');

disp([' ']);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-(minutes*60);
disp(['Elapsed time ',num2str(minutes),' minutes, ',num2str(seconds),' seconds']);
disp('Normal Termination of Program AVERAGE_DISPLACEMENT.M');
disp(['*****************************************************************'
]);
```

APPENDIX B3

```
%  CAPACITY_DIMENSION.M
%
%  This function determines the capacity dimension of a data
%  set by covering the three-dimensional data set with three-
%  dimensional volume elements of various lengths.  As the
%  lengths of the volume elements are increased, the number of
%  elements required to cover the entire data set is recorded.
%  The number of elements and the sizes of the elements are
%  then used to calculate the Capacity Dimension of the data
%  set.
%
%  XYZ->        time series from mat file
%  dt->         sampling time from mat file
%  trans_time-> length of transience to remove (seconds)
%  data_length-> number of data points to be examined
%
%  [cap_dim]=capacity_dimension(XYZ,dt,trans_time,data_length)
%
%  Copyright (c) 2002-2003 Andrew Dick

function [cap_dim]=capacity_dimension(XYZ,dt,trans_time,data_length);

%  Display Program Name, Date, and Time Information
%  ************************************************
tic;
disp(['*****************************************************************'
]);
disp(['Program to calculate Capacity Dimension by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp([' ']);

%  Preprocess Data
%  ************************************************
L=length(XYZ);
trans=trans_time/dt+1;
if L<data_length+trans-1;
  end_data=L;
else
  end_data=data_length+trans-1;
end;
time_series=XYZ(trans:end_data,:);
[N,n]=size(time_series);
```

```
r_attractor=0;
for i=1:n;
  r_attractor=r_attractor+(max(XYZ(:,i))-min(XYZ(:,i)))^2;
end;
r_attractor=sqrt(r_attractor);
rmin=0.01*r_attractor;
rmax=0.40*r_attractor;
rpts=50;
logr=linspace(log(rmin),log(rmax),rpts);
R=exp(logr);

disp(['Data previously iterated using 5th order Lie Series approximation']);
disp(['Data set consisted of ',num2str(N),' data points with a time step of ',num2str(dt),'
seconds']);
disp(['A transient period of ',num2str((trans-1)*dt),' seconds was removed']);
disp([' ']);

%  Determine Maximum and Minimum Values in Time Series
%  ****************************************************
for i=1:n;
  mx(i,1)=ceil(max(time_series(:,i)));
  mn(i,1)=floor(min(time_series(:,i)));
end;

%  Main Program
%  ****************************************************
clr=length(R);
for cl=1:clr;
  cube_max1=mn(1,1);
  cube_min1=mn(1,1);
  number_cubes=0;
  cube_length=R(cl);
%  Count number of volume elements that contain points
%  ****************************************************
  while cube_min1<mx(1,1);
    cube_max1=cube_max1+cube_length;
    cube_min2=mn(2,1);
    cube_max2=mn(2,1);
    while cube_min2<mx(2,1);
      cube_max2=cube_max2+cube_length;
      cube_min3=mn(3,1);
      cube_max3=mn(3,1);
      while cube_min3<mx(3,1);
        cube_max3=cube_max3+cube_length;
        index=1;
```

```
        while index<N;
          if time_series(index,1)>cube_min1;
            if time_series(index,1)<cube_max1;
              if time_series(index,2)>cube_min2;
                if time_series(index,2)<cube_max2;
                  if time_series(index,3)>cube_min3;
                    if time_series(index,3)<cube_max3;
                      number_cubes=number_cubes+1;
                      index=N;
                    end;
                  end;
                end;
              end;
            end;
          end;
          index=index+1;
        end;
        cube_min3=cube_max3;
      end;
      cube_min2=cube_max2;
    end;
    cube_min1=cube_max1;
  end;
  cube_lengths(cl)=1/(cube_length);
  number_of_cubes(cl)=number_cubes;
end;

%       Determine Range of Correlation Sums to be Examined
%  ********************************************************
finish=0;
L3=length(number_of_cubes);
for i=1:L3;
  if number_of_cubes(i)~=1;
    finish=finish+1;
  end;
  log_num_cube(i)=log(number_of_cubes(i));
end;
start=1;

%       Determine Correlation Dimension
%  ********************************************************
save capnums.mat cube_lengths log_num_cube;
[a,b,xs,ys,R]=linear_regression(log(cube_lengths),log_num_cube,start,finish);
cap_dim=a;

disp(['The Capacity Dimension of the data set was determined to be ',num2str(a)]);
```

```
disp([' ']);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-(minutes*60);
disp(['Elapsed time ',num2str(minutes),' minutes, ',num2str(seconds),' seconds']);
disp('Normal Termination of Program CAPACITY_DIMENSION.M');
disp(['************************************************************'
]);
```

APPENDIX B4

```
%  CORRELATION_DIMENSION.M
%
%  This function determines the correlation dimension of a data
%  set by examining the number of points within a given radius
%  of the other points.  As the size of the radius is varied,
%  the number of points within that distance of another point
%  are recorded.  The relationship between the number of points
%  and the length of the radius is then used to calculate the
%  Correlation Dimension of the data set.
%
%  XYZ->        time series from mat file
%  dt->         sampling time from mat file
%  trans_time-> length of transience to remove (seconds)
%  num_points-> number of data points to be examined
%  random->     number of data points selected at random
%
%  [cor_dim]=correlation_dimension_v3(XYZ,dt,trans_time,num_points,random)
%
%  Copyright (c) 2002-2003 Andrew Dick

function [cor_dim]=correlation_dimension_v3(XYZ,dt,trans_time,num_points,random);

%       Preprocess Data
%  ******************************************************
L1=length(XYZ);
conv=trans_time/dt+1;
if L1<num_points-1+conv;
  data_end=L1;
else;
  data_end=num_points-1+conv;
end;
xyz=XYZ(conv:data_end,:);
[N,m]=size(xyz);

r_attractor=0;
for i=1:m;
  r_attractor=r_attractor+(max(XYZ(:,i))-min(XYZ(:,i)))^2;
end;
r_attractor=sqrt(r_attractor);
rmin=0.005*r_attractor;
rmax=0.95*r_attractor;
rpts=50;
logr=linspace(log(rmin),log(rmax),rpts);
```

```
R=exp(logr);

W=10;

%  Display Program Name, Date, and Time Information
%  ************************************************
tic;
disp(['*****************************************************************'
]);
disp(['Correlation Dimension calculation program by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp(' ');
disp(['Data previously iterated using 5th order Lie Series approximation']);
disp(['First ',num2str(trans_time),' seconds of transient trajectory removed']);
disp(['Calculated using ',num2str(N),' data points']);
disp(['Calculated using a Theiler coefficient of W=',num2str(W)]);

%  Determine Correlation Sum
%  *************************************************
L2=length(R);
for rR=1:L2;
  total=0;
  for i=1:random;
    ii=0;
    while ii<=0;
      ii=floor(rand*N);
      if ii<W+2;
        ii=0;
      elseif ii>N-W-1;
        ii=0;
      end;
    end;
    for j=1:ii-W;
      sum=0;
      for k=1:m;
        sum=sum+(xyz(ii,k)-xyz(j,k))^2;
      end;
      r_sum=sqrt(sum);
      if r_sum<R(rR);
        total=total+1;
      end;
    end;
    for j=ii+W:N;
      sum=0;
      for k=1:m;
```

```
        sum=sum+(xyz(ii,k)-xyz(j,k))^2;
      end;
      r_sum=sqrt(sum);
      if r_sum<R(rR);
        total=total+1;
      end;
    end;
  end;
  Radius(rR,1)=R(rR);
  corr_sum(rR,1)=(1/((N-2*W+1)*random))*total;
end;

%       Determine Range of Correlation Sums to be Examined
%  ****************************************************
start=1;
finish=-1;
for i=1:L2;
  if corr_sum(i)==0;
    log_corr_sum(i)=-15;
  else;
    log_corr_sum(i)=log(corr_sum(i));
  end;
  if log_corr_sum(i)<-8.75;
    start=start+1;
    finish=finish+1;
  elseif log_corr_sum(i)<-0.5;
    finish=finish+1;
  end;
end;

%       Determine Correlation Dimension
%  ****************************************************
save corsums.mat Radius log_corr_sum;
[a,b,xs,ys,R]=linear_regression(log(Radius),log_corr_sum,start,finish);
cor_dim=a;

disp([' ']);
disp(['The Correlation Dimension of the data set was determined to be ',num2str(a)]);
disp([' ']);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-(minutes*60);
disp(['Elapsed time ',num2str(minutes),' minutes ',num2str(seconds),' seconds']);
disp(['Normal termination of CORRELATION_DIMENSION.M']);
disp(['************************************************************'
]);
```

APPENDIX B5

```
%  EMBEDDING_DIMENSION.M
%
%  This function determines the embedding dimension that should
%  be used when embedding a time series to reconstruct the
%  strange attractor for the the system from which the data was
%  collected.
%
%  XYZ->        time series from mat file
%  dt->         sampling time from mat file
%  tau->        delay value to reconstruct attractor
%  R->          distance from points to examine
%  pts->        number of points to randomly examine
%  em_high->    highest embedding dimension to examine
%  trans_time-> length of transience to remove (seconds)
%  data_length-> number of data points to be examined
%
%
[emb_dim]=embedding_dimension(XYZ,dt,tau,R,pts,em_high,trans_time,data_length)
%
%  Copyright (c) 2002-2003 Andrew Dick

function
[emb_dim]=embedding_dimension(XYZ,dt,tau,R,pts,em_high,trans_time,data_length);

%  Display Program Name, Date, and Time Information
%  ************************************************
tic;
disp(['*************************************************************************'
]);
disp(['Calculation of Embedding Dimension by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp([' ']);

%       Preprocess Data
%       ***********************************************
Lx=length(XYZ);
trans=trans_time/dt+1;
if Lx<data_length-1+trans;
  data_end=Lx;
else;
  data_end=data_length-1+trans;
end;
```

```
x=XYZ(trans:data_end,1);
L=length(x);
delay=floor(tau/dt);

%       Main Program
%       ************************************************
for em_dim=3:em_high;
%       Reconstruct Vectors from Data
%       ************************************************
        clear X;
        for dim=1:em_dim;
    for point=1:(L-em_dim*delay);
      X(point,dim,(em_dim-2))=x(point+(dim-1)*delay);
    end;
  end;

%       Determine Average Number of Points within
%       R of the Number of Random Points Being Examined
%       ************************************************
  total=0;
  for i=1:pts;
    ii=0;
    while ii<=0;
      ii=floor(rand*(L-em_dim*delay));
    end;
    sum=0;
    for j=1:(L-em_dim*delay);
      if ii~=j;
        r_sub=0;
        for k=1:em_dim;
          r_sub=r_sub+(X(ii,k,(em_dim-2))-X(j,k,(em_dim-2)))^2;
        end;
        r=sqrt(r_sub);
        if r<R;
          sum=sum+1;
        end;
      end;
    end;
    sum;
    total=total+sum;
  end;
  ave(em_dim-2)=total/pts;
  embed(em_dim-2)=em_dim;
end;

%       Display Algorithm Time, Workspace, and Plot
```

```
%      ************************************************
figure;
plot(embed,ave,'k.');
title(['Average Number of Data Points within ',num2str(R),' units of points']);
xlabel('Embedding Dimension');
ylabel('Average Number of Data Points');

L=length(ave);
for i=1:L-1;
  d_ave(i)=ave(i)-ave(i+1);
end;
%figure;
%plot([3:L+1],d_ave,'.k');
emb_dim=7;

disp(['To remove a sufficient amount of false nearest neighbors,']);
disp(['   an embedding dimension of ',num2str(emb_dim),' is required']);
disp(' ');
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-(minutes*60);
disp(['Elapsed time ',num2str(minutes),' minutes, ',num2str(seconds),' seconds']);
disp('Normal Termination of Program EMBEDDING_DIMENSION.M');
disp(['************************************************************'
]);
```

APPENDIX B6

```
%  INFORMATION_DIMENSION.M
%
%  This function determines the Information Dimension of the
%  data set being examined.  This is done by examining a relationship
%  between a complex entropy equation and the size of the divisions
%  into which the data is separated.  The function examines the
%  probability that the data points will fall into each one and
%  two-dimensional elements.
%
%  XYZ->        time series to be examined
%  dt->         time increment for evolution of system
%  trans_time-> length of transience removed, seconds
%  data_length-> number of data points to be examined
%
%  [info_dim]=information_dimension(XYZ,dt,trans_time,data_length)
%
%  Copyright (c) 2002-2003 Andrew Dick

function [info_dim]=information_dimension(XYZ,dt,trans_time,data_length);

%  Display Program Name, Date, and Time Information
%  ************************************************
tic;
disp(['*****************************************************************'
]);
disp(['Program to calculate Information Dimension by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp([' ']);

%  Preprocess Data
%  ***********************************************************
L=length(XYZ);
trans=trans_time/dt+1;
if L<data_length+trans-1;
  end_data=L;
else
  end_data=data_length+trans-1;
end;
time_series=XYZ(trans:end_data,:);
size_time=size(time_series);
N=max(size_time);
n=min(size_time);
```

```
r_attractor=0;
for i=1:n;
  r_attractor=r_attractor+(max(XYZ(:,i))-min(XYZ(:,i)))^2;
end;
r_attractor=sqrt(r_attractor);
rmin=0.01*r_attractor;
rmax=0.40*r_attractor;
rpts=50;
logr=linspace(log(rmin),log(rmax),rpts);
R=exp(logr);

disp(['Data previously iterated using 5th order Lie Series approximation']);
disp(['Data set consisted of ',num2str(N),' data points with a time step of ',num2str(dt),'
seconds']);
disp(['A transient period of ',num2str((trans-1)*dt),' seconds was removed']);
disp([' ']);

%  Determine Maximum and Minimum Values in Time Series
%  ****************************************************
for i=1:n;
  mx(i,1)=ceil(max(time_series(:,i)));
  mn(i,1)=floor(min(time_series(:,i)));
end;

%  Main Program
%  ****************************************************
clr=length(R);
for cl=1:clr;
  cube_max1=mn(1,1);
  cube_min1=mn(1,1);
  cube_length=R(cl);
  info_sum=0;
  pt_total=0;
%  Count number of points in each volume elements
%  ****************************************************
  while cube_min1<mx(1,1);
    cube_max1=cube_max1+cube_length;
    cube_min2=mn(2,1);
    cube_max2=mn(2,1);
    while cube_min2<mx(2,1);
      cube_max2=cube_max2+cube_length;
      cube_min3=mn(3,1);
      cube_max3=mn(3,1);
      while cube_min3<mx(3,1);
        cube_max3=cube_max3+cube_length;
```

```
        index=1;
        num_pt=0;
        while index<N;
          if time_series(index,1)>=cube_min1;
            if time_series(index,1)<cube_max1;
              if time_series(index,2)>=cube_min2;
                if time_series(index,2)<cube_max2;
                  if time_series(index,3)>=cube_min3;
                    if time_series(index,3)<cube_max3;
                      num_pt=num_pt+1;
                    end;
                  end;
                end;
              end;
            end;
          end;
          index=index+1;
        end;
        pt_total=pt_total+num_pt;
%  Summation of entropy for non-zero probabilities
%  ***************************************************
        if num_pt>0;
          prob_pt=num_pt/N;
          info_sum=info_sum-prob_pt*log(prob_pt);
        end;
        cube_min3=cube_max3;
      end;
      cube_min2=cube_max2;
    end;
    cube_min1=cube_max1;
  end;
  cube_lengths(cl)=1/cube_length;
  information_sum(cl)=info_sum;
end;

%       Determine Range of Correlation Sums to be Examined
%  ***************************************************
start=1;
finish=-1;
L3=length(information_sum);
for i=1:L3;
  if information_sum(i)>2;
    finish=finish+1;
  end;
end;
```

```
%       Determine Correlation Dimension
%  ***************************************************
save infosums.mat cube_lengths information_sum;
[a,b,xs,ys,R]=linear_regression(log(cube_lengths),information_sum,start,finish);
info_dim=a;

disp(['The Information Dimension of the data set was determined to be ',num2str(a)]);
disp([' ']);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-(minutes*60);
disp(['Elapsed time ',num2str(minutes),' minutes, ',num2str(seconds),' seconds']);
disp('Normal Termination of Program INFORMATION_DIMENSION.M');
disp(['************************************************************'
]);
```

APPENDIX B7

```
%  JACOBIAN_LORENZ.M
%
%  Function to determine the Jacobian matrix for the Lie Series
%  approximation of the Lorenz system
%
%  [DF]=jacobian_lorenz(X,h,sigma,b,r)
%
%  Copyright (c) 2002-2003 Andrew Dick

function [DF]=jacobian_lorenz(X,h,sigma,b,r);

x1=X(1);
x2=X(2);
x3=X(3);

h2=h^2/2;
h3=h^3/6;
h4=h^4/24;
h5=h^5/120;

%  ****************************************************************
U1xdx=-sigma;
U1xdy=sigma;
U1xdz=0;

U1ydx=r-x3;
U1ydy=-1;
U1ydz=-x1;

U1zdx=x2;
U1zdy=x1;
U1zdz=-b;

%  ****************************************************************
U2xdx=sigma^2+sigma*(r-x3);
U2xdy=-sigma^2-sigma;
U2xdz=-sigma*x1;

U2ydx=-sigma*(r-x3)-r+x3-2*x1*x2+b*x3;
U2ydy=sigma*(r-x3)+1-x1^2;
U2ydz=-sigma*(x2-x1)+x1+x1*b;
```

```
U2zdx=-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2;
U2zdy=sigma*(x2-x1)+x2*sigma-x1-x1*b;
U2zdz=-x1^2+b^2;

% ************************************************************
U3xdx=-(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-sigma)*(r-x3)-sigma*(x1*x2-b*x3)-
sigma*x1*x2;
U3xdy=(sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-sigma*x1^2;
U3xdz=-sigma^2*(x2-x1)-(-sigma^2-sigma)*x1+sigma*x1*b;

U3ydx=-2*x2*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-2*x1*(x1*r-
x2-x1*x3)+(sigma*(r-x3)+1-x1^2)*(r-x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-
x1)+x1+x1*b)*x2;
U3ydy=-2*x1*sigma*(x2-x1)+(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-
x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x1;
U3ydz=(sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-
x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b;

U3zdx=(2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-
b*x2)*sigma+(-sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-
x3)-2*x1*(x1*x2-b*x3)+(-x1^2+b^2)*x2;
U3zdy=(-sigma-1-b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-
b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-
x1^2+b^2)*x1;
U3zdz=-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b;

% ************************************************************
U4xdx=-2*x2*sigma^2*(x2-x1)-(-(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-sigma)*(r-
x3)-sigma*(x1*x2-b*x3)-sigma*x1*x2)*sigma-2*sigma*x1*(x1*r-x2-
x1*x3)+((sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-sigma*x1^2)*(r-
x3)+(2*sigma^2+sigma+sigma*b)*(x1*x2-b*x3)+(-sigma^2*(x2-x1)-(-sigma^2-
sigma)*x1+sigma*x1*b)*x2;
U4xdy=-2*sigma^2*x1*(x2-x1)+(-(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-
sigma)*(r-x3)-sigma*(x1*x2-b*x3)-sigma*x1*x2)*sigma-(sigma^2+sigma*(r-
x3))*sigma-sigma^2-sigma+sigma*x1^2-sigma^2*(x1*x2-b*x3)+(-sigma^2*(x2-x1)-(-
sigma^2-sigma)*x1+sigma*x1*b)*x1;
U4xdz=(2*sigma^2+sigma+sigma*b)*sigma*(x2-x1)-sigma^2*(x1*r-x2-x1*x3)-
((sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-sigma*x1^2)*x1-(-sigma^2*(x2-x1)-(-
sigma^2-sigma)*x1+sigma*x1*b)*b;

U4ydx=(4*x2*sigma-2*x1*r+2*x2+2*x1*x3-4*x1*(r-
x3)+2*(sigma+1+b)*x2)*sigma*(x2-x1)-(-2*x2*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-
2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-x1*x3)+(sigma*(r-x3)+1-x1^2)*(r-
x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x2)*sigma+(-
3*sigma*(x2-x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*(x1*r-x2-
x1*x3)+(-2*x1*sigma*(x2-x1)+(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-
```

```
x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x1)*(r-x3)+(-
(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-(sigma+1+b)*b)*(x1*x2-
b*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-
x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*x2;
U4ydy=(-3*sigma*(x2-x1)+2*sigma*x1-
3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*sigma*(x2-x1)+(-2*x2*sigma*(x2-x1)-(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-x1*x3)+(sigma*(r-x3)+1-
x1^2)*(r-x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x2)*sigma-
6*sigma*x1*(x1*r-x2-x1*x3)+2*x1*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-
2*x1*x2+b*x3)*sigma+sigma*(r-x3)+1-x1^2+sigma*(x1*x2-b*x3)-(-sigma*(x2-
x1)+x1+x1*b)*x1+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*x2-
b*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-
x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*x1;
U4ydz=(-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-(sigma+1+b)*b)*sigma*(x2-
x1)+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*r-x2-x1*x3)-(-2*x1*sigma*(x2-x1)+(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-
sigma*(x2-x1)+x1+x1*b)*x1)*x1+2*sigma*x1*(x1*x2-b*x3)-
((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-x1^2)*x1-(-
sigma*(x2-x1)+x1+x1*b)*b)*b;

U4zdx=(-2*(2*r-2*x3)*sigma+2*(-sigma-1-b)*(r-x3)-6*x1*x2+2*b*x3)*sigma*(x2-
x1)-((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+(-
sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-
b*x3)+(-x1^2+b^2)*x2)*sigma+(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-
x3)+sigma+1+b-3*x1^2+b^2)*(x1*r-x2-x1*x3)+((-sigma-1-b)*sigma*(x2-x1)+(-
x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-
sigma*(x2-x1)-x2*sigma+x1+x1*b+(-x1^2+b^2)*x1)*(r-x3)+(-3*sigma*(x2-
x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*(x1*x2-b*x3)+(-
2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*x2;
U4zdy=(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-
3*x1^2+b^2)*sigma*(x2-x1)+((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-
x1*x3+x1*(r-x3)-b*x2)*sigma+(-sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-
x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-b*x3)+(-x1^2+b^2)*x2)*sigma+(2*(-
sigma-1-b)*sigma-4*sigma)*(x1*r-x2-x1*x3)-(-sigma-1-b)*sigma*(x2-x1)-(-
x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma-2*sigma*(x1*r-x2-
x1*x3)+sigma*(x2-x1)+x2*sigma-x1-x1*b-(-x1^2+b^2)*x1-4*sigma*x1*(x1*x2-
b*x3)+(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-
x1^2+b^2)*b)*x1;
U4zdz=(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-
x2*sigma+x1+3*x1*b)*sigma*(x2-x1)-4*sigma*x1*(x1*r-x2-x1*x3)-((-sigma-1-
b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+2*sigma*(x1*r-
x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-x1^2+b^2)*x1)*x1-(-2*x1*sigma*(x2-
x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*b;

% ************************************************************
```

U5xdx=(4*x2*sigma^2-2*sigma*(x1*r-x2-x1*x3)-4*sigma*x1*(r-
x3)+2*(2*sigma^2+sigma+sigma*b)*x2)*sigma*(x2-x1)-(-2*x2*sigma^2*(x2-x1)-(-
(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-sigma)*(r-x3)-sigma*(x1*x2-b*x3)-
sigma*x1*x2)*sigma-2*sigma*x1*(x1*r-x2-x1*x3)+((sigma^2+sigma*(r-
x3))*sigma+sigma^2+sigma-sigma*x1^2)*(r-x3)+(2*sigma^2+sigma+sigma*b)*(x1*x2-
b*x3)+(-sigma^2*(x2-x1)-(-sigma^2-sigma)*x1+sigma*x1*b)*x2)*sigma+(-
3*sigma^2*(x2-x1)+2*sigma^2*x1-
3*x2*sigma^2+2*sigma*x1+(2*sigma^2+sigma+sigma*b)*x1-(-sigma^2-
sigma)*x1+sigma*x1*b)*(x1*r-x2-x1*x3)+(-2*sigma^2*x1*(x2-x1)+(-
(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-sigma)*(r-x3)-sigma*(x1*x2-b*x3)-
sigma*x1*x2)*sigma-(sigma^2+sigma*(r-x3))*sigma-sigma^2-sigma+sigma*x1^2-
sigma^2*(x1*x2-b*x3)+(-sigma^2*(x2-x1)-(-sigma^2-sigma)*x1+sigma*x1*b)*x1)*(r-
x3)+(-(2*sigma^2+sigma+sigma*b)*sigma-sigma^2*(r-x3)+3*sigma*x1^2-
(sigma^2+sigma*(r-x3))*sigma-sigma^2-sigma-
(2*sigma^2+sigma+sigma*b)*b)*(x1*x2-
b*x3)+((2*sigma^2+sigma+sigma*b)*sigma*(x2-x1)-sigma^2*(x1*r-x2-x1*x3)-
((sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-sigma*x1^2)*x1-(-sigma^2*(x2-x1)-(-
sigma^2-sigma)*x1+sigma*x1*b)*b)*x2;
U5xdy=(-3*sigma^2*(x2-x1)+2*sigma^2*x1-
3*x2*sigma^2+2*sigma*x1+(2*sigma^2+sigma+sigma*b)*x1-(-sigma^2-
sigma)*x1+sigma*x1*b)*sigma*(x2-x1)+(-2*x2*sigma^2*(x2-x1)-(-
(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-sigma)*(r-x3)-sigma*(x1*x2-b*x3)-
sigma*x1*x2)*sigma-2*sigma*x1*(x1*r-x2-x1*x3)+((sigma^2+sigma*(r-
x3))*sigma+sigma^2+sigma-sigma*x1^2)*(r-x3)+(2*sigma^2+sigma+sigma*b)*(x1*x2-
b*x3)+(-sigma^2*(x2-x1)-(-sigma^2-sigma)*x1+sigma*x1*b)*x2)*sigma-
6*sigma^2*x1*(x1*r-x2-x1*x3)+2*sigma^2*x1*(x2-x1)-(-(sigma^2+sigma*(r-
x3))*sigma+(-sigma^2-sigma)*(r-x3)-sigma*(x1*x2-b*x3)-
sigma*x1*x2)*sigma+(sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-
sigma*x1^2+sigma^2*(x1*x2-b*x3)-(-sigma^2*(x2-x1)-(-sigma^2-
sigma)*x1+sigma*x1*b)*x1+((2*sigma^2+sigma+sigma*b)*sigma+sigma^2+sigma^2*
b)*(x1*x2-b*x3)+((2*sigma^2+sigma+sigma*b)*sigma*(x2-x1)-sigma^2*(x1*r-x2-
x1*x3)-((sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-sigma*x1^2)*x1-(-
sigma^2*(x2-x1)-(-sigma^2-sigma)*x1+sigma*x1*b)*b)*x1;
U5xdz=(-(2*sigma^2+sigma+sigma*b)*sigma-sigma^2*(r-x3)+3*sigma*x1^2-
(sigma^2+sigma*(r-x3))*sigma-sigma^2-sigma-
(2*sigma^2+sigma+sigma*b)*b)*sigma*(x2-
x1)+((2*sigma^2+sigma+sigma*b)*sigma+sigma^2+sigma^2*b)*(x1*r-x2-x1*x3)-(-
2*sigma^2*x1*(x2-x1)+(-(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-sigma)*(r-x3)-
sigma*(x1*x2-b*x3)-sigma*x1*x2)*sigma-(sigma^2+sigma*(r-x3))*sigma-sigma^2-
sigma+sigma*x1^2-sigma^2*(x1*x2-b*x3)+(-sigma^2*(x2-x1)-(-sigma^2-
sigma)*x1+sigma*x1*b)*x1)*x1+2*sigma^2*x1*(x1*x2-b*x3)-
((2*sigma^2+sigma+sigma*b)*sigma*(x2-x1)-sigma^2*(x1*r-x2-x1*x3)-
((sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-sigma*x1^2)*x1-(-sigma^2*(x2-x1)-(-
sigma^2-sigma)*x1+sigma*x1*b)*b)*b;

```
U5ydx=((-6*r+6*x3)*sigma*(x2-x1)-2*(4*x2*sigma-2*x1*r+2*x2+2*x1*x3-4*x1*(r-
x3)+2*(sigma+1+b)*x2)*sigma+(6*sigma+4+2*b)*(x1*r-x2-x1*x3)+2*(-3*sigma*(x2-
x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*(r-x3)+6*x1*(x1*x2-
b*x3)+2*(-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*x2)*sigma*(x2-x1)-((4*x2*sigma-2*x1*r+2*x2+2*x1*x3-4*x1*(r-
x3)+2*(sigma+1+b)*x2)*sigma*(x2-x1)-(-2*x2*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-
2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-x1*x3)+(sigma*(r-x3)+1-x1^2)*(r-
x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x2)*sigma+(-
3*sigma*(x2-x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*(x1*r-x2-
x1*x3)+(-2*x1*sigma*(x2-x1)+(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-
x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x1)*(r-x3)+(-
(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-(sigma+1+b)*b)*(x1*x2-
b*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-
x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*x2)*sigma+((6*sigma+4+2*b)*sigma*(x2-
x1)-(-3*sigma*(x2-x1)+2*sigma*x1-
3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*sigma+(4*x2*sigma-
2*x1*r+2*x2+2*x1*x3-4*x1*(r-x3)+2*(sigma+1+b)*x2)*sigma-7*sigma*(x1*r-x2-
x1*x3)-6*sigma*x1*(r-x3)+3*sigma*(x2-x1)-2*sigma*x1+3*x2*sigma-3*x1-
(sigma+1+b)*x1-x1*b+((sigma+1+b)*sigma+sigma+sigma*b)*x2+(-
(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*x1+(sigma+1+b)*sigma*(x2-x1)-(sigma*(r-x3)+1-x1^2)*x1-(-
sigma*(x2-x1)+x1+x1*b)*b)*(x1*r-x2-x1*x3)+((-3*sigma*(x2-x1)+2*sigma*x1-
3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*sigma*(x2-x1)+(-2*x2*sigma*(x2-x1)-(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-x1*x3)+(sigma*(r-x3)+1-
x1^2)*(r-x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x2)*sigma-
6*sigma*x1*(x1*r-x2-x1*x3)+2*x1*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-
2*x1*x2+b*x3)*sigma+sigma*(r-x3)+1-x1^2+sigma*(x1*x2-b*x3)-(-sigma*(x2-
x1)+x1+x1*b)*x1+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*x2-
b*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-
x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*x1)*(r-x3)+(8*x1*sigma*(x2-x1)-(-
(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*sigma+((sigma+1+b)*sigma+sigma+sigma*b)*(r-x3)-(-3*sigma*(x2-
x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*x1-(-sigma*(r-x3)-r+x3-
2*x1*x2+b*x3)*sigma+sigma*(r-x3)+1-x1^2+3*sigma*(x1*x2-b*x3)-(-sigma*(x2-
x1)+x1+x1*b)*x1+2*sigma*x1*x2-(-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*b)*(x1*x2-b*x3)+((-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*sigma*(x2-x1)+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*r-x2-
x1*x3)-(-2*x1*sigma*(x2-x1)+(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-
x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-sigma*(x2-
x1)+x1+x1*b)*x1)*x1+2*sigma*x1*(x1*x2-b*x3)-((sigma+1+b)*sigma*(x2-x1)-
sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-x1^2)*x1-(-sigma*(x2-
x1)+x1+x1*b)*b)*b)*x2;
U5ydy=((6*sigma+4+2*b)*sigma*(x2-x1)-(-3*sigma*(x2-x1)+2*sigma*x1-
3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*sigma+(4*x2*sigma-
2*x1*r+2*x2+2*x1*x3-4*x1*(r-x3)+2*(sigma+1+b)*x2)*sigma-7*sigma*(x1*r-x2-
x1*x3)-6*sigma*x1*(r-x3)+3*sigma*(x2-x1)-2*sigma*x1+3*x2*sigma-3*x1-
```

(sigma+1+b)*x1-x1*b+((sigma+1+b)*sigma+sigma+sigma*b)*x2+(-
(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*x1+(sigma+1+b)*sigma*(x2-x1)-(sigma*(r-x3)+1-x1^2)*x1-(-
sigma*(x2-x1)+x1+x1*b)*b)*sigma*(x2-x1)+((4*x2*sigma-2*x1*r+2*x2+2*x1*x3-
4*x1*(r-x3)+2*(sigma+1+b)*x2)*sigma*(x2-x1)-(-2*x2*sigma*(x2-x1)-(-sigma*(r-x3)-
r+x3-2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-x1*x3)+(sigma*(r-x3)+1-x1^2)*(r-
x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x2)*sigma+(-
3*sigma*(x2-x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*(x1*r-x2-
x1*x3)+(-2*x1*sigma*(x2-x1)+(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-
x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x1)*(r-x3)+(-
(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-(sigma+1+b)*b)*(x1*x2-
b*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-
x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*x2)*sigma+(-6*sigma^2*(x2-x1)+2*(-
3*sigma*(x2-x1)+2*sigma*x1-
3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*sigma+12*sigma*x1+2*((sigma+1+b)*sigm
a+sigma+sigma*b)*x1)*(x1*r-x2-x1*x3)-(-3*sigma*(x2-x1)+2*sigma*x1-
3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*sigma*(x2-x1)-(-2*x2*sigma*(x2-x1)-(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-x1*x3)+(sigma*(r-x3)+1-
x1^2)*(r-x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-
x1)+x1+x1*b)*x2)*sigma+6*sigma*x1*(x1*r-x2-x1*x3)-2*x1*sigma*(x2-x1)+(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-
sigma*(x2-x1)+x1+x1*b)*x1-((sigma+1+b)*sigma+sigma+sigma*b)*(x1*x2-b*x3)-
((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-x1^2)*x1-(-
sigma*(x2-x1)+x1+x1*b)*b)*x1+((-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*sigma-(sigma+1+b)*sigma-sigma-sigma*b+8*sigma*x1^2-
((sigma+1+b)*sigma+sigma+sigma*b)*b)*(x1*x2-b*x3)+((-(sigma+1+b)*sigma-
2*sigma*(r-x3)+3*x1^2-1-(sigma+1+b)*b)*sigma*(x2-
x1)+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*r-x2-x1*x3)-(-2*x1*sigma*(x2-x1)+(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-
sigma*(x2-x1)+x1+x1*b)*x1)*x1+2*sigma*x1*(x1*x2-b*x3)-
((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-x1^2)*x1-(-
sigma*(x2-x1)+x1+x1*b)*b)*b)*x1;
U5ydz=(8*x1*sigma*(x2-x1)-(-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*sigma+((sigma+1+b)*sigma+sigma+sigma*b)*(r-x3)-(-3*sigma*(x2-
x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*x1-(-sigma*(r-x3)-r+x3-
2*x1*x2+b*x3)*sigma+sigma*(r-x3)+1-x1^2+3*sigma*(x1*x2-b*x3)-(-sigma*(x2-
x1)+x1+x1*b)*x1+2*sigma*x1*x2-(-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*b)*sigma*(x2-x1)+((-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-
(sigma+1+b)*b)*sigma-(sigma+1+b)*sigma-sigma-sigma*b+8*sigma*x1^2-
((sigma+1+b)*sigma+sigma+sigma*b)*b)*(x1*r-x2-x1*x3)-((-3*sigma*(x2-
x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*sigma*(x2-x1)+(-
2*x2*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-
x1*x3)+(sigma*(r-x3)+1-x1^2)*(r-x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-
x1)+x1+x1*b)*x2)*sigma-6*sigma*x1*(x1*r-x2-x1*x3)+2*x1*sigma*(x2-x1)-(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma+sigma*(r-x3)+1-x1^2+sigma*(x1*x2-b*x3)-
(-sigma*(x2-x1)+x1+x1*b)*x1+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*x2-

b*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*x1)*x1+(2*sigma^2*(x2-x1)-2*((sigma+1+b)*sigma+sigma+sigma*b)*x1-4*sigma*x1*b)*(x1*x2-b*x3)-((-(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-(sigma+1+b)*b)*sigma*(x2-x1)+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*r-x2-x1*x3)-(-2*x1*sigma*(x2-x1)+(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x1)*x1+2*sigma*x1*(x1*x2-b*x3)-((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*b)*b;

U5zdx=(-6*x2*sigma*(x2-x1)-2*(-2*(2*r-2*x3)*sigma+2*(-sigma-1-b)*(r-x3)-6*x1*x2+2*b*x3)*sigma-6*x1*(x1*r-x2-x1*x3)+2*(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*(r-x3)+(6*sigma+2+4*b)*(x1*x2-b*x3)+2*(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*x2)*sigma*(x2-x1)-((-2*(2*r-2*x3)*sigma+2*(-sigma-1-b)*(r-x3)-6*x1*x2+2*b*x3)*sigma*(x2-x1)-((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+(-sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-b*x3)+(-x1^2+b^2)*x2)*sigma+(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*(x1*r-x2-x1*x3)+((-sigma-1-b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-x1^2+b^2)*x1)*(r-x3)+(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*(x1*x2-b*x3)+(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*x2)*sigma+(-8*x1*sigma*(x2-x1)-(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*sigma+(-2*(2*r-2*x3)*sigma+2*(-sigma-1-b)*(r-x3)-6*x1*x2+2*b*x3)*sigma+(2*(-sigma-1-b)*sigma-4*sigma)*(r-x3)+(-sigma-1-b)*sigma-(2*r-2*x3)*sigma-2*sigma*(r-x3)-sigma-1-b+3*x1^2-b^2-4*sigma*(x1*x2-b*x3)-4*sigma*x1*x2+(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*x1-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*(x1*r-x2-x1*x3)+((-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*sigma*(x2-x1)+((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+(-sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-b*x3)+(-x1^2+b^2)*x2)*sigma+(2*(-sigma-1-b)*sigma-4*sigma)*(x1*r-x2-x1*x3)-(-sigma-1-b)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma-2*sigma*(x1*r-x2-x1*x3)+sigma*(x2-x1)+x2*sigma-x1-x1*b-(-x1^2+b^2)*x1-4*sigma*x1*(x1*x2-b*x3)+(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*x1)*(r-x3)+((6*sigma+2+4*b)*sigma*(x2-x1)-(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*sigma-6*sigma*(x1*r-x2-x1*x3)-4*sigma*x1*(r-x3)-(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*x1-(-sigma-1-b)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+sigma*(x2-x1)+x2*sigma-x1-x1*b-(-x1^2+b^2)*x1-(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*b)*(x1*x2-b*x3)+((-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*sigma*(x2-x1)-4*sigma*x1*(x1*r-x2-x1*x3)-((-sigma-1-

b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-x1^2+b^2)*x1)*x1-(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*b)*x2;
U5zdy=(-8*x1*sigma*(x2-x1)-(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*sigma+(-2*(2*r-2*x3)*sigma+2*(-sigma-1-b)*(r-x3)-6*x1*x2+2*b*x3)*sigma+(2*(-sigma-1-b)*sigma-4*sigma)*(r-x3)+(-sigma-1-b)*sigma-(2*r-2*x3)*sigma-2*sigma*(r-x3)-sigma-1-b+3*x1^2-b^2-4*sigma*(x1*x2-b*x3)-4*sigma*x1*x2+(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*x1-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*sigma*(x2-x1)+((-2*(2*r-2*x3)*sigma+2*(-sigma-1-b)*(r-x3)-6*x1*x2+2*b*x3)*sigma*(x2-x1)-((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+(-sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-b*x3)+(-x1^2+b^2)*x2)*sigma+(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*(x1*r-x2-x1*x3)+((-sigma-1-b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-x1^2+b^2)*x1)*(r-x3)+(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*(x1*x2-b*x3)+(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*x2)*sigma+(2*(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*sigma-4*(-sigma-1-b)*sigma+8*sigma-8*sigma*x1^2)*(x1*r-x2-x1*x3)-(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*sigma*(x2-x1)-((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+(-sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-b*x3)+(-x1^2+b^2)*x2)*sigma-(2*(-sigma-1-b)*sigma-4*sigma)*(x1*r-x2-x1*x3)+(-sigma-1-b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-x1^2+b^2)*x1+4*sigma*x1*(x1*x2-b*x3)-(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*x1+(-4*sigma^2*(x2-x1)+(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*sigma+4*sigma*x1-(2*(-sigma-1-b)*sigma-4*sigma)*x1+4*sigma*x1*b)*(x1*x2-b*x3)+((-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*sigma*(x2-x1)-4*sigma*x1*(x1*r-x2-x1*x3)-((-sigma-1-b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-x1^2+b^2)*x1)*x1-(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*b)*x1;
U5zdz=((6*sigma+2+4*b)*sigma*(x2-x1)-(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*sigma-6*sigma*(x1*r-x2-x1*x3)-4*sigma*x1*(r-x3)-(-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*x1-(-sigma-1-b)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+sigma*(x2-x1)+x2*sigma-x1-x1*b-(-x1^2+b^2)*x1-(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*b)*sigma*(x2-x1)+(-4*sigma^2*(x2-x1)+(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*sigma+4*sigma*x1-(2*(-sigma-1-b)*sigma-4*sigma)*x1+4*sigma*x1*b)*(x1*r-x2-x1*x3)-((-(-sigma-1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*sigma*(x2-x1)+((2*r-

```
2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+(-sigma-1-
b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-b*x3)+(-
x1^2+b^2)*x2)*sigma+(2*(-sigma-1-b)*sigma-4*sigma)*(x1*r-x2-x1*x3)-(-sigma-1-
b)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma-2*sigma*(x1*r-
x2-x1*x3)+sigma*(x2-x1)+x2*sigma-x1-x1*b-(-x1^2+b^2)*x1-4*sigma*x1*(x1*x2-
b*x3)+(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-
x1^2+b^2)*b)*x1)*x1+8*sigma*x1^2*(x1*x2-b*x3)-((-3*sigma*(x2-x1)+2*sigma*x1-
(-sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*sigma*(x2-x1)-4*sigma*x1*(x1*r-x2-x1*x3)-
((-sigma-1-b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-
b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-
x1^2+b^2)*x1)*x1-(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-
x1^2+b^2)*b)*b)*b;

% ************************************************************
DF=zeros(3,3);
DF(1,1)=1+U1xdx*h+U2xdx*h2+U3xdx*h3+U4xdx*h4+U5xdx*h5;
DF(1,2)=U1xdy*h+U2xdy*h2+U3xdy*h3+U4xdy*h4+U5xdy*h5;
DF(1,3)=U1xdz*h+U2xdz*h2+U3xdz*h3+U4xdz*h4+U5xdz*h5;
DF(2,1)=U1ydx*h+U2ydx*h2+U3ydx*h3+U4ydx*h4+U5ydx*h5;
DF(2,2)=1+U1ydy*h+U2ydy*h2+U3ydy*h3+U4ydy*h4+U5ydy*h5;
DF(2,3)=U1ydz*h+U2ydz*h2+U3ydz*h3+U4ydz*h4+U5ydz*h5;
DF(3,1)=U1zdx*h+U2zdx*h2+U3zdx*h3+U4zdx*h4+U5zdx*h5;
DF(3,2)=U1zdy*h+U2zdy*h2+U3zdy*h3+U4zdy*h4+U5zdy*h5;
DF(3,3)=1+U1zdz*h+U2zdz*h2+U3zdz*h3+U4zdz*h4+U5zdz*h5;
```

APPENDIX B8

```
%  LARGEST_LYAPUNOV.M
%
%  This function determines the largest Lyapunov exponent for
%  an unkown system.  The function computes the value from
%  a time series of data collected from the system.
%
%  XYZ->        time series from mat file
%  dt->         sampling time from mat file
%  trans_time-> length of transience to remove
%  data_length-> number of data points
%  period->     approx length of time for one revolution of the attractor (sec)
%  incr->       number of time increments to monitor divergence per cycle
%  cycles->     number of cycle used to calculate exponent
%
%
[larg_lyp,lyp_ave]=largest_lyapunov(XYZ,dt,trans_time,data_length,period,incr,cycles)
%
%  Copyright (c) 2002-2003 Andrew Dick

function
[larg_lyp,lyp_ave]=largest_lyapunov(XYZ,dt,trans_time,data_length,period,incr,cycles);

%  Display Program Name, Date, and Time Information
%  ************************************************
tic;
disp(['*****************************************************************'
]);
disp(['Largest Lyapunov exponent calculation program by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp(' ');

%       Preprocess data
%  ************************************************
base=2;
trans=floor(trans_time/dt)+1;
XYZ=XYZ(trans:trans-1+data_length,:);
per=floor(period/dt);
clear time;
[L,m]=size(XYZ);
sum=0;

disp(['Data previously iterated using 5th order Lie Series approximation']);
```

```
disp(['First ',num2str(trans_time),' seconds of transient trajectory removed']);
disp(['Calculated from ',num2str(L),' data points with time step of ',num2str(dt),'
seconds']);
disp(['Calculated using ',num2str(cycles),' ',num2str(incr*dt),' seconds segments']);

%       Main program
%  ************************************************
for n=1:cycles;
  step=(n-1)*incr+1;
  L2=L-step-incr;
  %     Determine distance between ref pt and all others
  %  ************************************************
  clear adjpt;
  adjpt=zeros(L2,1);
  for i=1:L2;
    adjptsq=0;
    for j=1:m;
      adjptsq=adjptsq+(XYZ(step,j)-XYZ(step+(i-1),j))^2;
    end;
    adjpt(i,1)=sqrt(adjptsq);
  end;
  dist_0=min(adjpt(per:L2,1));
  %     Locate closest point to ref pt
  %  ************************************************
  index=per;
  while adjpt(index,1)~=dist_0;
    index=index+1;
  end;
  adj_i=index+step-1;
  %     Determine distance between points after increment
  %  ************************************************
  distsq=0;
  for j=1:m;
    distsq=distsq+(XYZ(step+incr,j)-XYZ(adj_i+incr,j))^2;
  end;
  dist_1=sqrt(distsq);
  sum=sum+log(dist_1/dist_0)/log(base);
  lyp_ave(n,1)=(1/(n*incr*dt))*sum;
end;

largest_exponent=(1/(cycles*incr*dt))*sum;
larg_lyp=largest_exponent;

disp(['The largest Lyapunov exponent for this data set is calculated to be
',num2str(largest_exponent)]);
disp(' ');
```

```
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-minutes*60;
disp(['Elapsed time ',num2str(minutes),' minutes ',num2str(seconds),' seconds']);
disp('Normal Termination of LARGEST_LYAPUNOV.M');
disp(['************************************************************'
]);
```

APPENDIX B9

```
%  LIE_APPROX.M
%
%  This function using lie series approximations of the Lorenz
%  System, Rossler System, Duffing Equation, and Van der Pol
%  Equation to determine the next step in the system.
%
%  system->     The system to be evolved
%               'lorenz___','rossler__','duffing__','vanderpol'
%  order->      The order of the lie series approximation: 1-5
%  X->          Initial conditions: [x0,y0,z0]
%  dt->         Desired time increment for evolution
%
%  [NX,DF]=lie_approx_r1(system,order,X,dt)
%
%  Copyright (c) 2002-2003 Andrew Dick

function [NX,DF]=lie_approx(system,order,X,dt);

x1=X(1,1);
x2=X(1,2);
x3=X(1,3);
h=dt;

h2=h^2/2;
h3=h^3/6;
h4=h^4/24;
h5=h^5/120;

U1x=0;U1y=0;U1z=0;
U2x=0;U2y=0;U2z=0;
U3x=0;U3y=0;U3z=0;
U4x=0;U4y=0;U4z=0;
U5x=0;U5y=0;U5z=0;

if order<=5;
    %  Lorenz System
    %  ***************************************************************
  elseif system=='lorenz1__';
    sigma=10;
    b=8/3;
    r=28;
    [DF]=jacobian_lorenz(X,h,sigma,b,r);
    U1x=sigma*(x2-x1);
```

```
    U1y=x1*r-x2-x1*x3;
    U1z=x1*x2-b*x3;
    if order>=2;
      U2x=(-sigma^2*(x2-x1)+sigma*(x1*r-x2-x1*x3));
      U2y=((r-x3)*sigma*(x2-x1)-x1*r+x2+x1*x3-x1*(x1*x2-b*x3));
      U2z=(x2*sigma*(x2-x1)+x1*(x1*r-x2-x1*x3)-b*(x1*x2-b*x3));
    end;
    if order>=3;
      U3x=((sigma^2+sigma*(r-x3))*sigma*(x2-x1)+(-sigma^2-sigma)*(x1*r-x2-
x1*x3)-sigma*x1*(x1*x2-b*x3));
      U3y=((-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma*(x2-x1)+(sigma*(r-x3)+1-
x1^2)*(x1*r-x2-x1*x3)+(-sigma*(x2-x1)+x1+x1*b)*(x1*x2-b*x3));
      U3z=((-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma*(x2-x1)+(sigma*(x2-
x1)+x2*sigma-x1-x1*b)*(x1*r-x2-x1*x3)+(-x1^2+b^2)*(x1*x2-b*x3));
    end;
    if order>=4;
      U4x=((-(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-sigma)*(r-x3)-sigma*(x1*x2-
b*x3)-sigma*x1*x2)*sigma*(x2-x1)+((sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-
sigma*x1^2)*(x1*r-x2-x1*x3)+(-sigma^2*(x2-x1)-(-sigma^2-
sigma)*x1+sigma*x1*b)*(x1*x2-b*x3));
      U4y=((-2*x2*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-
2*x1*(x1*r-x2-x1*x3)+(sigma*(r-x3)+1-x1^2)*(r-x3)+(sigma+1+b)*(x1*x2-b*x3)+(-
sigma*(x2-x1)+x1+x1*b)*x2)*sigma*(x2-x1)+(-2*x1*sigma*(x2-x1)+(-sigma*(r-x3)-
r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-sigma*(x2-
x1)+x1+x1*b)*x1)*(x1*r-x2-x1*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-
x1*x3)-(sigma*(r-x3)+1-x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*(x1*x2-b*x3));
      U4z=(((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-
b*x2)*sigma+(-sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-
x3)-2*x1*(x1*x2-b*x3)+(-x1^2+b^2)*x2)*sigma*(x2-x1)+((-sigma-1-b)*sigma*(x2-
x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-
sigma*(x2-x1)-x2*sigma+x1+x1*b+(-x1^2+b^2)*x1)*(x1*r-x2-x1*x3)+(-
2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*(x1*x2-
b*x3));
    end;
    if order==5;
      U5x=((-2*x2*sigma^2*(x2-x1)-(-(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-
sigma)*(r-x3)-sigma*(x1*x2-b*x3)-sigma*x1*x2)*sigma-2*sigma*x1*(x1*r-x2-
x1*x3)+((sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-sigma*x1^2)*(r-
x3)+(2*sigma^2+sigma+sigma*b)*(x1*x2-b*x3)+(-sigma^2*(x2-x1)-(-sigma^2-
sigma)*x1+sigma*x1*b)*x2)*sigma*(x2-x1)+(-2*sigma^2*x1*(x2-x1)+(-
(sigma^2+sigma*(r-x3))*sigma+(-sigma^2-sigma)*(r-x3)-sigma*(x1*x2-b*x3)-
sigma*x1*x2)*sigma-(sigma^2+sigma*(r-x3))*sigma-sigma^2-sigma+sigma*x1^2-
sigma^2*(x1*x2-b*x3)+(-sigma^2*(x2-x1)-(-sigma^2-
sigma)*x1+sigma*x1*b)*x1)*(x1*r-x2-
x1*x3)+((2*sigma^2+sigma+sigma*b)*sigma*(x2-x1)-sigma^2*(x1*r-x2-x1*x3)-
```

```
((sigma^2+sigma*(r-x3))*sigma+sigma^2+sigma-sigma*x1^2)*x1-(-sigma^2*(x2-x1)-(-
sigma^2-sigma)*x1+sigma*x1*b)*b)*(x1*x2-b*x3));
      U5y=(((4*x2*sigma-2*x1*r+2*x2+2*x1*x3-4*x1*(r-
x3)+2*(sigma+1+b)*x2)*sigma*(x2-x1)-(-2*x2*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-
2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-x1*x3)+(sigma*(r-x3)+1-x1^2)*(r-
x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x2)*sigma+(-
3*sigma*(x2-x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*(x1*r-x2-
x1*x3)+(-2*x1*sigma*(x2-x1)+(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-
x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-sigma*(x2-x1)+x1+x1*b)*x1)*(r-x3)+(-
(sigma+1+b)*sigma-2*sigma*(r-x3)+3*x1^2-1-(sigma+1+b)*b)*(x1*x2-
b*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-
x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*x2)*sigma*(x2-x1)+((-3*sigma*(x2-
x1)+2*sigma*x1-3*x2*sigma+3*x1+(sigma+1+b)*x1+x1*b)*sigma*(x2-x1)+(-
2*x2*sigma*(x2-x1)-(-sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-2*x1*(x1*r-x2-
x1*x3)+(sigma*(r-x3)+1-x1^2)*(r-x3)+(sigma+1+b)*(x1*x2-b*x3)+(-sigma*(x2-
x1)+x1+x1*b)*x2)*sigma-6*sigma*x1*(x1*r-x2-x1*x3)+2*x1*sigma*(x2-x1)-(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma+sigma*(r-x3)+1-x1^2+sigma*(x1*x2-b*x3)-
(-sigma*(x2-x1)+x1+x1*b)*x1+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*x2-
b*x3)+((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-
x1^2)*x1-(-sigma*(x2-x1)+x1+x1*b)*b)*x1)*(x1*r-x2-x1*x3)+((-(sigma+1+b)*sigma-
2*sigma*(r-x3)+3*x1^2-1-(sigma+1+b)*b)*sigma*(x2-
x1)+((sigma+1+b)*sigma+sigma+sigma*b)*(x1*r-x2-x1*x3)-(-2*x1*sigma*(x2-x1)+(-
sigma*(r-x3)-r+x3-2*x1*x2+b*x3)*sigma-sigma*(r-x3)-1+x1^2-sigma*(x1*x2-b*x3)+(-
sigma*(x2-x1)+x1+x1*b)*x1)*x1+2*sigma*x1*(x1*x2-b*x3)-
((sigma+1+b)*sigma*(x2-x1)-sigma*(x1*r-x2-x1*x3)-(sigma*(r-x3)+1-x1^2)*x1-(-
sigma*(x2-x1)+x1+x1*b)*b)*b)*(x1*x2-b*x3));
      U5z=(((-2*(2*r-2*x3)*sigma+2*(-sigma-1-b)*(r-x3)-
6*x1*x2+2*b*x3)*sigma*(x2-x1)-((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-
x1*x3+x1*(r-x3)-b*x2)*sigma+(-sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-
x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-b*x3)+(-x1^2+b^2)*x2)*sigma+(-(-sigma-
1-b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*(x1*r-x2-
x1*x3)+((-sigma-1-b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-
b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-
x1^2+b^2)*x1)*(r-x3)+(-3*sigma*(x2-x1)+2*sigma*x1-(-sigma-1-b)*x1-
x2*sigma+x1+3*x1*b)*(x1*x2-b*x3)+(-2*x1*sigma*(x2-x1)-(sigma*(x2-
x1)+x2*sigma-x1-x1*b)*x1-(-x1^2+b^2)*b)*x2)*sigma*(x2-x1)+((-(-sigma-1-
b)*sigma+(2*r-2*x3)*sigma+2*sigma*(r-x3)+sigma+1+b-3*x1^2+b^2)*sigma*(x2-
x1)+((2*r-2*x3)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma+(-
sigma-1-b)*(x1*r-x2-x1*x3)+(sigma*(x2-x1)+x2*sigma-x1-x1*b)*(r-x3)-2*x1*(x1*x2-
b*x3)+(-x1^2+b^2)*x2)*sigma+(2*(-sigma-1-b)*sigma-4*sigma)*(x1*r-x2-x1*x3)-(-
sigma-1-b)*sigma*(x2-x1)-(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-b*x2)*sigma-
2*sigma*(x1*r-x2-x1*x3)+sigma*(x2-x1)+x2*sigma-x1-x1*b-(-x1^2+b^2)*x1-
4*sigma*x1*(x1*x2-b*x3)+(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-
x1*b)*x1-(-x1^2+b^2)*b)*x1)*(x1*r-x2-x1*x3)+((-3*sigma*(x2-x1)+2*sigma*x1-(-
sigma-1-b)*x1-x2*sigma+x1+3*x1*b)*sigma*(x2-x1)-4*sigma*x1*(x1*r-x2-x1*x3)-((-
sigma-1-b)*sigma*(x2-x1)+(-x2*sigma+x1*r-x2-x1*x3+x1*(r-x3)-
```

```
b*x2)*sigma+2*sigma*(x1*r-x2-x1*x3)-sigma*(x2-x1)-x2*sigma+x1+x1*b+(-
x1^2+b^2)*x1)*x1-(-2*x1*sigma*(x2-x1)-(sigma*(x2-x1)+x2*sigma-x1-x1*b)*x1-(-
x1^2+b^2)*b)*b)*(x1*x2-b*x3));
    end;
end;

% Next Iteration Values
%      ************************************************************
nx1=x1+U1x*h+U2x*h2+U3x*h3+U4x*h4+U5x*h5;
nx2=x2+U1y*h+U2y*h2+U3y*h3+U4y*h4+U5y*h5;
nx3=x3+U1z*h+U2z*h2+U3z*h3+U4z*h4+U5z*h5;

NX=[nx1,nx2,nx3];
```

APPENDIX B10

```
%  LIE_SERIES_ITERATION.M
%
%  This program calls upon the function containing the Lie series
%  approximations of four different systems to create data sets.
%
%  system->     system to be iterated
%               'lorenz___','rossler__','duffing__','vanderpol'
%  order->      order of lie series approximation
%  dt->         time increment for evolution of system
%  size->       magnitude of added noise
%  T->          length of data set, seconds
%  X->          initial conditions, [x0,y0,z0]
%
%  lie_series_iteratoin(system,order,dt,size,T,X)
%
%  Copyright (c) 2002-2003 Andrew Dick

%  Parameters
%  ********************************************************
%system='linear___';
system='lorenz1__';
%system='lorenz2__';
%system='rossler__';
%system='duffing__';
%system='vanderpol';
order=5;
dt=0.01;
size=0.005;
T=30;
X=[1,0,0];

%  Display Program Name, Date, and Time Information
%  ********************************************************
tic
disp(['*****************************************************************']
;
disp(['Lie Series approximation iteration program by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp(' ');

%       Preprocessing
%  ************************************************
```

```
L=T/dt;
XYZ=zeros(L+1,3);
XYZ(1,:)=X;
time=[0:dt:T]';

disp('Iteration Parameters');
disp([' ',num2str(order),'th order approximation of ',system,' system']);
disp([' Time step of ',num2str(dt),' seconds']);
disp([' Magnitude of noise added: ',num2str(size)]);
disp([' Length of iteration ',num2str(T),' seconds']);
disp([' Initial conditions: ',num2str(X(1)),',',num2str(X(2)),',',num2str(X(3))]);

%       Iteration of data
%  ************************************************
for n=1:L;
  [NX,DF]=lie_approx(system,order,X,dt);
  XYZ(n+1,:)=NX+size*(rand-0.5);
  X=NX;
end;

%       Plot of three time series created by itereation
%  ************************************************
L=length(XYZ);
time=[dt:dt:dt*L];
figure;plot(time(501:1000),XYZ(501:1000,1));
title('Lorenz System with Classical Parameters');
xlabel('Time');ylabel('X Variable');
for i=1:3;
  subplot(3,1,i);
  plot(time,XYZ(:,i));
end;

%       Save iterated data and time step to .mat file
%  ************************************************
save data000.mat XYZ dt

%       Plot 2-D and 3-D projections of attractor
%  ************************************************
figure;
subplot(2,2,1);
plot(XYZ(:,1),XYZ(:,2));
subplot(2,2,2);
plot(XYZ(:,1),XYZ(:,3));
subplot(2,2,3);
plot(XYZ(:,2),XYZ(:,3));
%subplot(2,2,4);
```

```
%plot3(XYZ(:,1),XYZ(:,2),XYZ(:,3));

disp(' ');
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-minutes*60;
disp(['Elapsed time ',num2str(minutes),' minutes ',num2str(seconds),' seconds']);
disp('Normal Termination of LIE_SERIES_ITERATION.M');
disp(['************************************************************']);
```

APPENDIX B11

```
%  LINEAR_REGRESSION.M
%
%  This function fits a linear curve to the two sets of
%  data that it reads.  The linear regression is done
%  using matrix multiplication and left matrix division.
%  The function only looks at a portion of the data
%  defined by two of the input values.  The result of
%  this function is a plot of the input data and the
%  curve fit line.  This function also calculates the
%  R-squared value corresponding to the linear fit.
%
%  x->     independent set of data
%  y->     dependent set of data
%  start-> value of data sets to start using for regression
%  finish-> value of data sets to stop using for regression
%
%  [a,b,xs,ys,R]=linear_regression(x,y,start,finish)
%
%  Copyright (c) 2002-2003 Andrew Dick

function [a,b,xs,ys,R]=linear_regression(x,y,start,finish);

%  Preprocess Data
%  ************************************************************
x=x(:);
y=y(:);

%  Select Portion of Data to Examine
%  ************************************************************
xsub=x(start:finish);
ysub=y(start:finish);
xsub=[xsub,ones(size(xsub))];

%  Multiply Vectors to Produce 2x2 and 2x1 matrices
%  ************************************************************
A=xsub'*xsub;
B=xsub'*ysub;

%  If Matrix A is not Singular, Determine Slope and Intercept
%  ************************************************************
if det(A)~=0;
  a=A\B;
else;
```

```
  a=[1 1]';
end;

%  Assign the Slope and Intercept to the Output Variables
%  Select Values to Plot Fit Curve
%  ************************************************************
b=a(2);                          %intercept
a=a(1);                          %slope
%xs=[xsub(1);xsub(max(size(xsub)))];
xs=xsub(:,1);
ys=a*xs+b;

xsum=sum(xsub(:,1));
ysum=sum(ysub);
xy=dot(xsub(:,1),ysub);
for j=1:finish-start+1;
  xsq(j)=xsub(j,1)^2;
  ysq(j)=ysub(j)^2;
end;
xsqsum=sum(xsq);
ysqsum=sum(ysq);
n=finish-start+1;
r_num=n*xy-xsum*ysum;
r_den=sqrt((n*xsqsum-xsum^2)*(n*ysqsum-ysum^2));
R=r_num/r_den;

%  Plot Original Data and Fit Curve
%  Linear Regression used to Determine Fractal Dimensions
%  ************************************************************
figure;
plot(x,y,'*k',xs,ys,'--r');
title(['Attractor dimension approximated to be ',num2str(a),' with Rsq: ',num2str(R^2)]);
```

APPENDIX B12

```
%  LYAPUNOV_SPECTRUM.M
%
%  This function determines the Lyapunov spectrum for a
%  known nonlinear system with parameter values that will
%  produce chaotic dynamics.  Once the Lyapunov spectrum
%  has been calculated, the Lyapunov Dimension is also
%  calculated.
%
%  SYSTEM->     system to be examined
%             'lorenz___','rossler__','duffing__','vanderpol'
%  ORDER->      order of Lie Series Approximation
%  X0->         initial conditions, [1,0,0]
%  DT->         time increment for evolution of system
%  T->          length of transient time before calculation
%  LOOPS->      number of loops with GSR process
%
%  [lyp_spec,lyp_dim]=lyapunov_spectrum_v2(SYSTEM,ORDER,X0,DT,T,LOOPS)
%
%  Copyright (c) 2002-2003 Andrew Dick

%function
[lyp_spec,lyp_dim]=lyapunov_spectrum_v2(SYSTEM,ORDER,X0,DT,T,LOOPS);
clear

SYSTEM='lorenz2__';
ORDER=5;
X0=[1,0,0];
DT=0.01;
T=10;
LOOPS=1*10^4;


N=3;
BASE=2;

%  Display Program Name, Date, and Time Information
%  ***************************************************
tic;
disp(['*****************************************************************'
]);
disp(['Lyapunov Spectrum calculation program by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
```

```
% Display Control Parameters
% ******************************************************
disp([' ']);
disp(['Control Parameters:']);
disp([' System Examined: ',SYSTEM]);
disp([' Initial Conditions: ',num2str(X0(1)),',',num2str(X0(2)),',',num2str(X0(3))]);
disp([' Time Increment: ',num2str(DT)]);
disp([' Transient Time: ',num2str(T)]);
disp([' Number of Iterations: ',num2str(LOOPS)]);

% Initialize Parameters
% ******************************************************
Nt=T/DT;
X=X0;
XYZ=zeros(Nt,3);
XYZ(1,:)=X;
LOOP=0;
V=zeros(N,N);

% Iterate System Through Transients
% ******************************************************
for n=1:Nt;
  [NX,DF]=lie_approx(SYSTEM,ORDER,X,DT);
  XYZ(n+1,:)=NX;
  X=XYZ(n+1,:);
end;

figure;
plot3(XYZ(:,1),XYZ(:,2),XYZ(:,3));
title('Verify Convergence to Attractor');
xlabel('x1');
ylabel('x2');
zlabel('x3');

disp([' ']);
disp(['System Iterated So Trajectory will Converge to the Attractor']);
disp(['Elapsed time ',num2str(toc),' seconds']);

T0_current=XYZ(Nt,1:3);
E=eye(N,N);
SUM=zeros(N,1);

while LOOP<LOOPS;
  LOOP=LOOP+1;
  % Further Iterate System
```

```
  %  ********************************************************
  [T0_next,DF]=lie_approx(SYSTEM,ORDER,T0_current,DT);
  T0_current=T0_next;
  % Map Basis Vectors;
  %  ********************************************************
  V=E*transpose(DF);
  %  Normalize First Vector
  %  ********************************************************
  ZNORM(1)=0;
  for I=1:N;
    ZNORM(1)=ZNORM(1)+V(1,I)^2;
  end;
  ZNORM(1)=sqrt(ZNORM(1));
  for I=1:N;
    E(1,I)=V(1,I)/ZNORM(1);
  end;
  %  Construct New Set of Vectors
  %  ********************************************************
  for J=2:N;
    for K=1:(J-1);
      GSC(K)=dot(V(J,:),E(K,:));
    end;
    for K=1:N;
      for L=1:(J-1);
        V(J,K)=V(J,K)-GSC(L)*E(L,K);
      end;
    end;
    %  Normalize
    %  ********************************************************
    ZNORM(J)=0;
    for K=1:N;
      ZNORM(J)=ZNORM(J)+V(J,K)^2;
    end;
    ZNORM(J)=sqrt(ZNORM(J));
    E(J,:)=V(J,:)/ZNORM(J);
  end;
  %  Update Log-Magnitudes
  for K=1:N;
    SUM(K)=SUM(K)+log(ZNORM(K))/log(BASE);
  end;
  EXPS(LOOP,:)=SUM'/(LOOP*DT);
end;
%  Calculate Lyapunov Exponents
%  ********************************************************
for K=1:N;
  ZEXP(K)=SUM(K)/(DT*LOOPS);
```

```
end;

figure;
hold on;
plot(EXPS(:,1),'b');
plot(EXPS(:,2),'k');
plot(EXPS(:,3),'r');
title('Lyapunov Exponents');
xlabel('Iterations');
ylabel('Exp3                    Exp2                Exp1');
hold off;

[lyp_spec,index]=sort(ZEXP);
lyp_spec=lyp_spec*[0 0 1;0 1 0;1 0 0];

lyp_dim=2+(lyp_spec(1)+lyp_spec(2))/abs(lyp_spec(3));

disp([' ']);
disp(['Analysis Results:']);
disp([' Lyapunov Exponent Spectrum:
',num2str(lyp_spec(1)),',',num2str(lyp_spec(2)),',',num2str(lyp_spec(3))]);
disp([' Lyapunov Dimension: ',num2str(lyp_dim)]);
disp([' ']);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-minutes*60;
disp(['Elapsed time ',num2str(minutes),' minutes ',num2str(seconds),' seconds']);
disp(['Normal termination of LYAPUNOV_SPECTRUM.M']);
disp(['************************************************************'
]);
```

APPENDIX B13

```
%  MUTUAL_INFORMATION.M
%
%  This function utilizes the Average Mutual Information
%  of the original data set and a range of delayed data
%  sets to determine the delay time value to be used for
%  attractor reconstruction.  The function choses a delay
%  time value when the the average mutual information
%  is at a local minimum.
%
%  XYZ->        data set to be examined
%  dt->         time increment of data set
%  numseg1->    number of divisions along the first axis
%  numseg2->    number of divisions along the second axis
%  taus->       the maximum delay value examined
%  trans_time-> length of transience removed, seconds
%  data_length-> number of data points to be examined
%
%
[mut_info_min]=mutual_information(XYZ,dt,numseg1,numseg2,taus,trans_time,data_le
ngth)
%
%  Copyright (c) 2002-2003 Andrew Dick

function
[mut_info_min]=mutual_information(XYZ,dt,numseg1,numseg2,taus,trans_time,data_le
ngth);

%  Display Program Name, Date, and Time Information
%  ************************************************
tic;
disp(['*****************************************************************'
]);
disp(['Delay Time value calculation program using Average Mutual Information by
Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
disp([' ']);

%  Preprocess Data
%  ************************************************
trans=trans_time/dt+1;
L=length(XYZ);
if L<trans-1+data_length
```

```
  data_end=L;
else;
  data_end=trans-1+data_length;
end;
X=XYZ(trans:data_end,1);
L1=length(X);
sums=zeros(taus+1,1);

disp(['Data set consisted of ',num2str(L1),' data points with a time step of ',num2str(dt),'
seconds']);
disp(['A transient period of ',num2str((trans-1)*dt),' seconds was removed']);
disp([' ']);

%  Main Program
%  ********************************************************
for tau=1:(taus+1);
%  Embedding Data with Delay Time
%  ********************************************************
  for i=1:(L1-tau+1);
    x1(i)=X(i);
    x2(i)=X(i+tau-1);
  end;
%  Determine Range of Data and Size of Increments
%  ********************************************************
  L2=length(x1);
  min1=floor(min(x1));
  max1=ceil(max(x1));
  range1=max1-min1;
  inc1=range1/numseg1;
  min2=floor(min(x2));
  max2=ceil(max(x2));
  range2=max2-min2;
  inc2=range2/numseg2;
  num=zeros(numseg1,numseg2);
  low1=min1;
  i=0;
%  Count Data Points in each Two-Dimensional Element
%  ********************************************************
  while low1<max1;
    high1=low1+inc1;
    i=i+1;
    low2=min2;
    j=0;
    while low2<max2;
      high2=low2+inc2;
      j=j+1;
```

```
      for i1=1:L2;
        if x1(i1)>low1;
          if x1(i1)<=high1;
            if x2(i1)>low2;
              if x2(i1)<=high2;
                num(j,i)=num(i,j)+1;
              end;
            end;
          end;
        end;
      end;
      low2=high2;
    end;
    low1=high1;
  end;
%  Calculate Two-Dimensional Probability
%  ******************************************************
  pxy=num*(1/L2);
%  Calculate One-Dimensional Probabilities
%  ******************************************************
  for xi=1:numseg1;
    px(1,xi)=sum(pxy(:,xi));
  end;
  for xj=1:numseg2;
    py(xj,1)=sum(pxy(xj,:));
  end;
%  Compute Average Mutual Information Sum
%  ******************************************************
  for i=1:numseg1;
    for j=1:numseg2;
      if pxy(j,i)>0;
        if px(i)>0;
          if py(j)>0;
            sums(tau)=sums(tau)+pxy(j,i)*log2(pxy(j,i)/(px(i)*py(j)));
          end;
        end;
      end;
    end;
  end;
  delay(tau)=(tau-1)*dt;
end;

%  Plot Average Mutual Information Versus Delay Time
%  ******************************************************
figure;
plot(delay,sums);
```

```
title('Average Mutual Information');
xlabel('Delay Time, sec');
ylabel('Amount of Average Mutual Information');

%  Locate and Plot Local Minimum
%  ****************************************************
diff=1;
index=0;
while diff>0;
  index=index+1;
  diff=sums(index)-sums(index+1);
end;
mutualinformation=index-1;mut_info_min=mutualinformation*dt;
hold on;
plot(delay(index),sums(index),'.k');
hold off;

disp(['Delay Value for First Local Minimum is ',num2str(mutualinformation*dt),'
seconds'])

%  Reconstruct and Plot Attractor using Local Minimum
%  ****************************************************
clear x1;
clear x2;
for i=1:L1-mutualinformation;
  x1(i)=X(i+mutualinformation);
  x2(i)=X(i);
end;

figure;
plot(x1,x2);
title(['Time Delay of ',num2str(mutualinformation*dt),' seconds']);
xlabel('Data from time series');
ylabel('Delayed data from time series');

disp([' ']);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-(minutes*60);
disp(['Elapsed time ',num2str(minutes),' minutes, ',num2str(seconds),' seconds']);
disp('Normal Termination of Program MUTUAL_INFORMATION.M');
disp(['************************************************************'
]);
```

APPENDIX B14

```
%  SINGULAR_SYSTEM_APPROACH.M
%
%  This function uses Singular Value Decomposition to examine a trajectory matrix
%  created by embedding a time series of a single variable with a delay time of
%  one time increment and an embedding dimension greater than required for the
%  system.  Singular Value Decomposition produces the singular vectors and
%  singular values for the trajectory matrix.  Examination of the singular values
%  provides information as to which singular vectors contain deterministic dominant
%  values and which contain noise dominant values.  The number of vectors containing
%  deterministic dominant values is the rank corresponding to the system.
%
%  XYZ->        time series from mat file
%  dt->         sampling time from mat file
%  trans_time-> transient time removed from signal(seconds)
%  data_length-> number of data points to be examined
%  tau->        delay time calculated for data set
%  rs->         resampling, rs times original rate
%  tau->        delay window, seconds
%
%  [A]=singular_system_approach_v2(XYZ,dt,trans_time,data_length,rs,tau)
%
%  Copyright (c) 2002-2003 Andrew Dick

function [A]=singular_system_approach_v2(XYZ,dt,trans_time,data_length,rs,tau);

%  Resample data
%  ******************************************************
Lw=length(XYZ);
wx=floor(Lw/rs);
for i=1:wx;
  xyz(i,1)=XYZ((i-1)*rs+1);
end;
clear XYZ;
XYZ=xyz;

%  Display Program Name, Date, and Time Information
%  ******************************************************
tic;
disp(['*************************************************************************'
]);
disp(['Singular System Approach for attractor reconstruction by Andrew Dick']);
disp([date]);
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
```

```
disp([' ']);

%  Preprocess Data
%  ******************************************************
n=floor(tau/dt);
L=length(XYZ);
trans=floor(trans_time/dt)+1;
if L<data_length+trans-1;
  data_end=L;
else
  data_end=data_length-1+trans;
end;
x=XYZ(trans:data_end,1);
Nt=length(x);
N=Nt-n+1;
disp([num2str(data_end-trans+1),' data points used with time step of ',num2str(dt*rs)]);
disp([num2str((trans-1)*dt),' seconds of transient data removed']);

%  Embed Data into Trajectory Matrix
%  ******************************************************
for i=1:N;
  for j=1:n;
    Xx(i,j)=x(i+j-1);
  end;
end;
X=(N)^(-0.5)*Xx;

%  Perform Singular Value Decomposition
%  Calculate Singular Values
%  ******************************************************
Y=X'*X;
[V,D]=eig(Y);
for j=1:n;
  sums=0;
  for i=1:N;
    dot_prod=dot(X(i,:),D(:,j));
    sums=sums+dot_prod^2;
  end;
  sigma(j)=((1/N)*sums)^(0.5);
end;
[sigma_1,index_1]=sort(sigma);
for j=1:n;
  sigma_2(j)=sigma_1(n+1-j);
  index_2(j)=index_1(n+1-j);
end;
sigma=sigma_2;
```

```
for j=1:n;
  C(:,j)=V(:,index_2(j));
end;

%  Determine Rank of Trajectory Matrix
%  ***************************************************
sum_sigma=sum(sigma);
for i=1:n;
  log_p(i)=log10(sigma(i)/sum_sigma);
  k(i)=i;
end;

%  Plot Singular Value Array
%  ***************************************************
figure;
plot(k,log_p,'-+b');
title('Analysis of Singular Values');
xlabel('Value of k');
ylabel('Log[sigma(k)/Sum(sigma)]');

%  Multiply Trajectory Matrix and Singular Vectors
%  Plot Reconstruction From Resulting Matrix
%  ***************************************************
A=X*C;
figure;
title('Reconstructed Attractor Using Singular System Approach');
subplot(2,2,1);plot(A(:,1),A(:,2));
xlabel('C1');ylabel('C2');
subplot(2,2,2);plot(A(:,1),A(:,3));
xlabel('C1');ylabel('C3');
subplot(2,2,3);plot(A(:,2),A(:,3));
xlabel('C2');ylabel('C3');
subplot(2,2,4);plot3(A(:,1),A(:,2),A(:,3));
xlabel('C1');ylabel('C2');zlabel('C3');

disp(' ');
t1=clock; disp([num2str(t1(4)),':',num2str(t1(5)),':',num2str(t1(6))]);
minutes=floor(toc/60);seconds=toc-(minutes*60);
disp(['Elapsed time ',num2str(minutes),' minutes, ',num2str(seconds),' seconds']);
disp('Normal Termination of Program SINGULAR_SYSTEM_APPROACH.M');
disp(['************************************************************'
]);
```

APPENDIX C1

****

Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
18:49:54.31

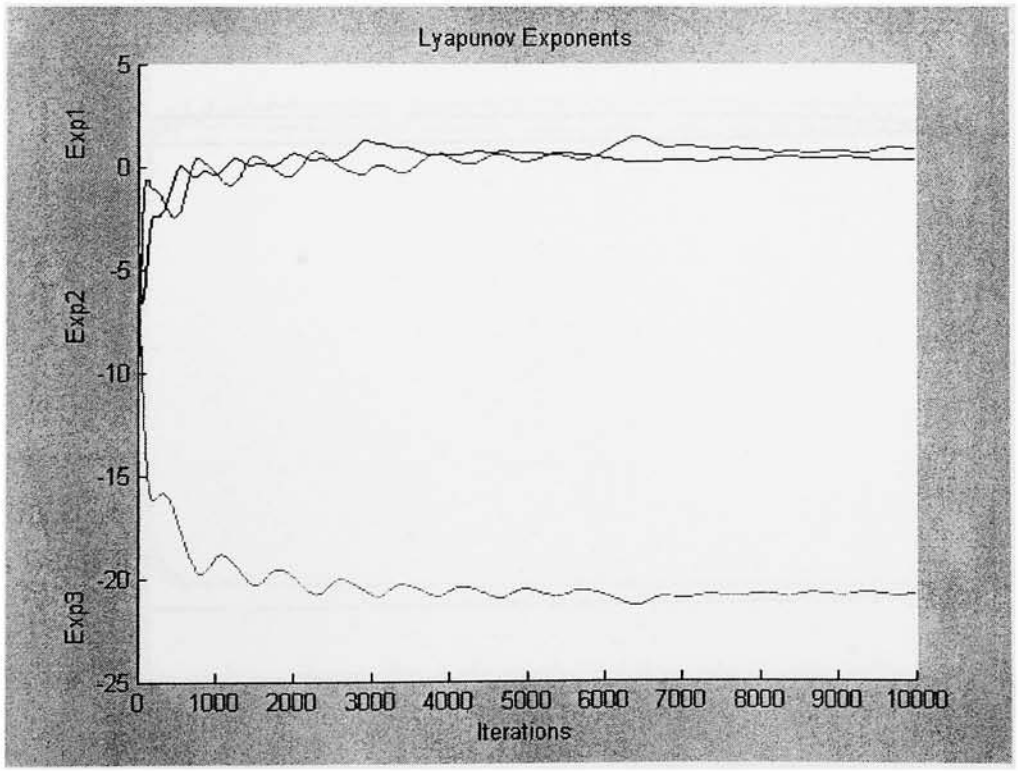
Control Parameters:
System Examined: Lorenz
Parameters: $\sigma = 10$, $r = 28$, $b = 8/3$
Initial Conditions: 1, 0, 0
Time Increment: 0.001
Transient Time: 20
Number of Iterations: 10000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 149.67 seconds

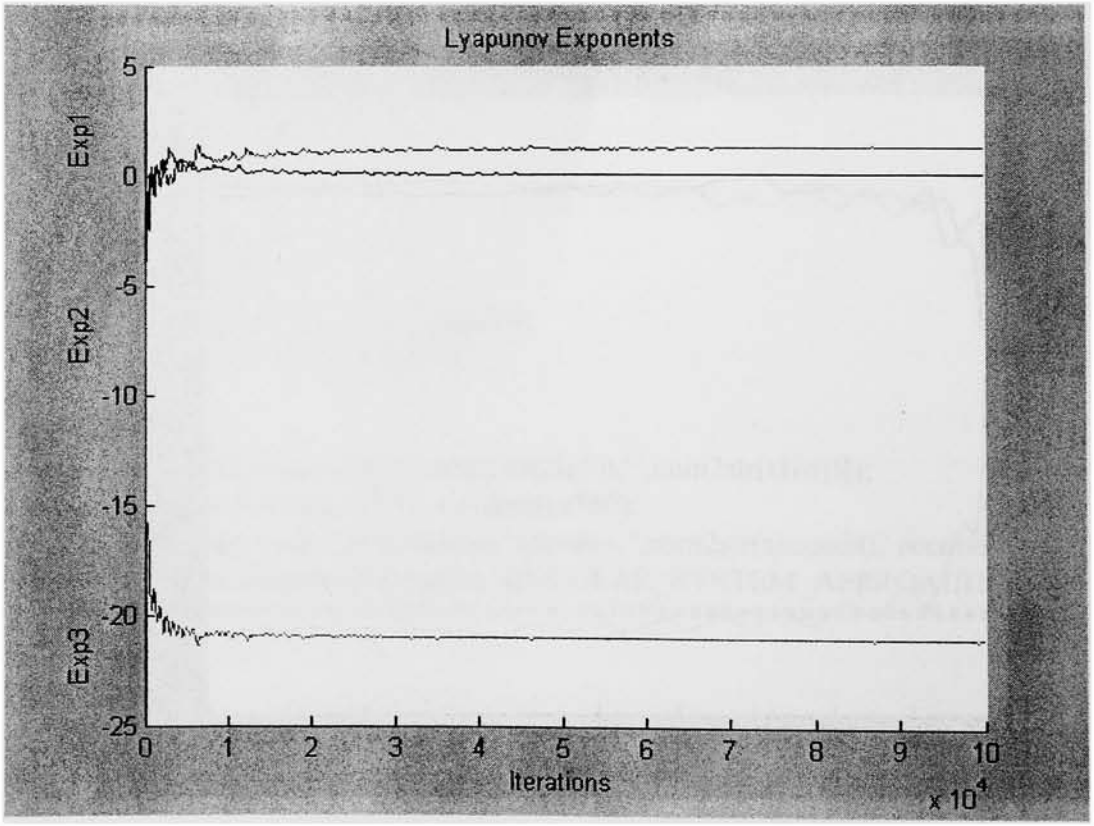
Analysis Results:
Lyapunov Exponent Spectrum: 0.79137, 0.28077, -20.789
Lyapunov Dimension: 2.0516

18:54:8.67
Elapsed time 4 minutes 14.36 seconds
Normal termination of LYAPUNOV_SPECTRUM.M

****



```
**********************************************************************
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
18:54:8.67

Control Parameters:
  System Examined: Lorenz
  Parameters: σ = 10, r = 28, b = 8/3
  Initial Conditions: 1, 0, 0
  Time Increment: 0.001
  Transient Time: 20
  Number of Iterations: 100000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 149.95 seconds

Analysis Results:
  Lyapunov Exponent Spectrum: 1.2867, 0.034343, -21.0379
  Lyapunov Dimension: 2.0628

19:41:31.01
Elapsed time 47 minutes 22.4 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
**********************************************************************
```



****

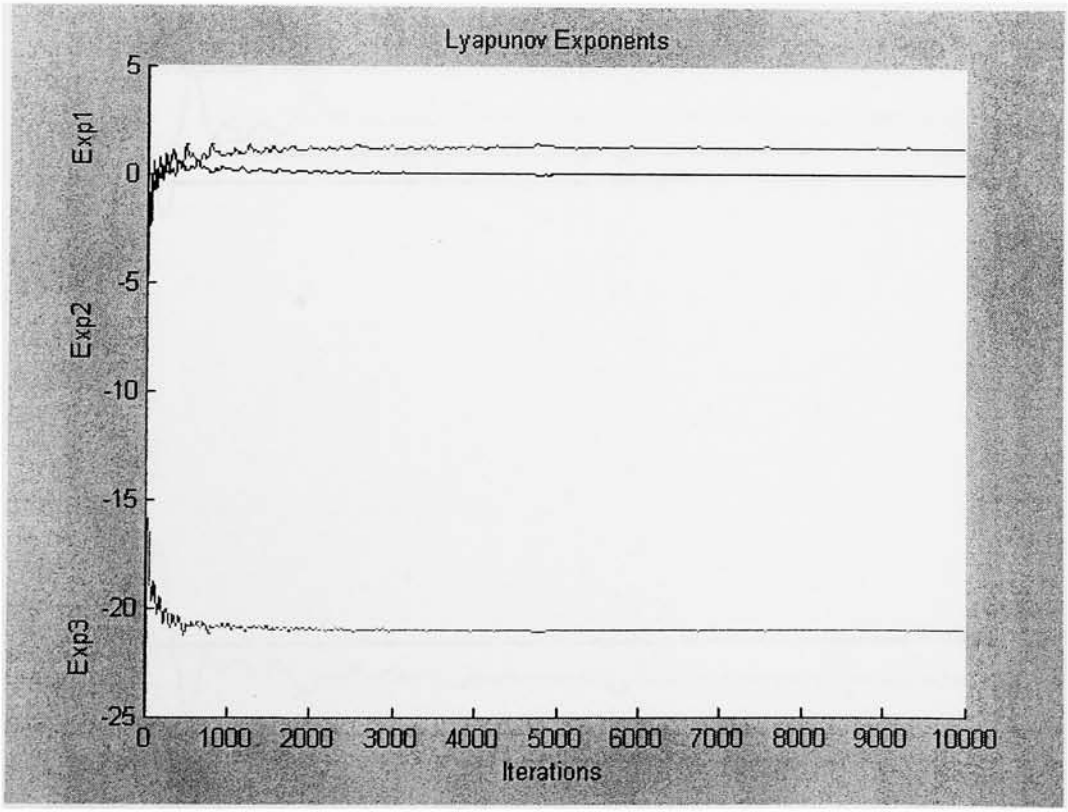
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
18:2:32.3

Control Parameters:
System Examined: Lorenz
Parameters: $\sigma = 10$, $r = 28$, $b = 8/3$
Initial Conditions: 1, 0, 0
Time Increment: 0.01
Transient Time: 20
Number of Iterations: 10000

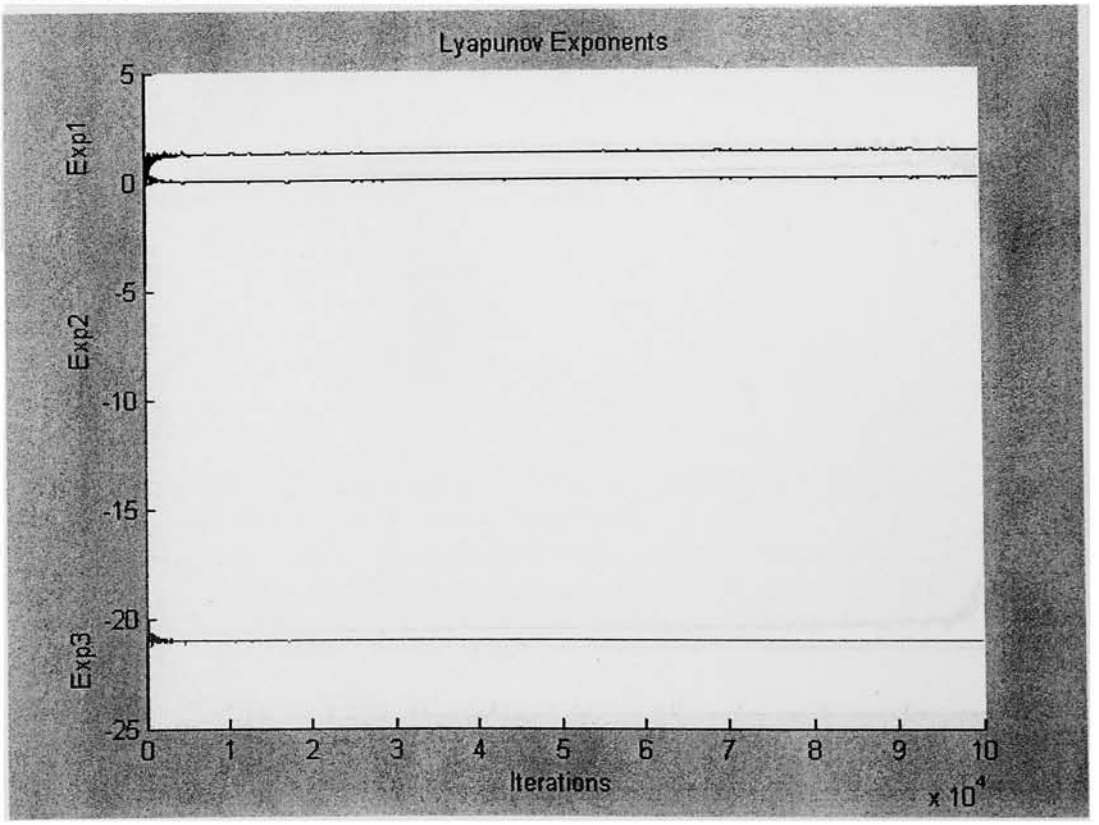
System Iterated So Trajectory will Converge to the Attractor
Elapsed time 15.27 seconds

Analysis Results:
Lyapunov Exponent Spectrum: 1.2876, 0.012108, -21.0166
Lyapunov Dimension: 2.0618

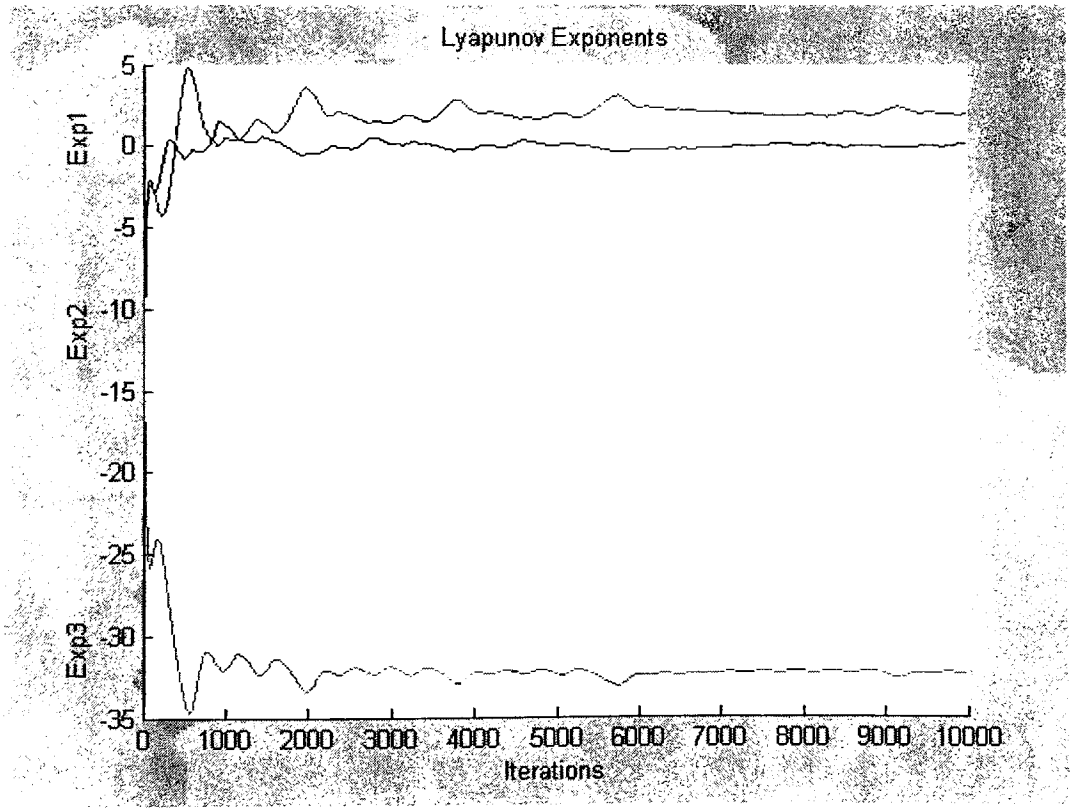
18:4:31.71
Elapsed time 1 minutes 59.41 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
****



```
*****************************************************************
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
18:4:31.71

Control Parameters:
  System Examined: Lorenz
  Parameters: σ = 10, r = 28, b = 8/3
  Initial Conditions: 1, 0, 0
  Time Increment: 0.01
  Transient Time: 20
  Number of Iterations: 100000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 15.05 seconds

Analysis Results:
  Lyapunov Exponent Spectrum: 1.3096, .0016219, -21.0281
  Lyapunov Dimension: 2.0624

18:49:54.31
Elapsed time 45 minutes 22.6 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
*****************************************************************
```



```
**********************************************************************
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
20:28:50.77

Control Parameters:
  System Examined: Lorenz
  Parameters: σ = 16, r = 45.92, b = 4
  Initial Conditions: 1, 0, 0
  Time Increment: 0.001
  Transient Time: 20
  Number of Iterations: 10000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 149.84 seconds

Analysis Results:
  Lyapunov Exponent Spectrum: 1.9695, 0.1128, -32.3789
  Lyapunov Dimension: 2.0643

20:33:5.29
Elapsed time 4 minutes 14.52 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
**********************************************************************
```



****

Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
20:33:5.29

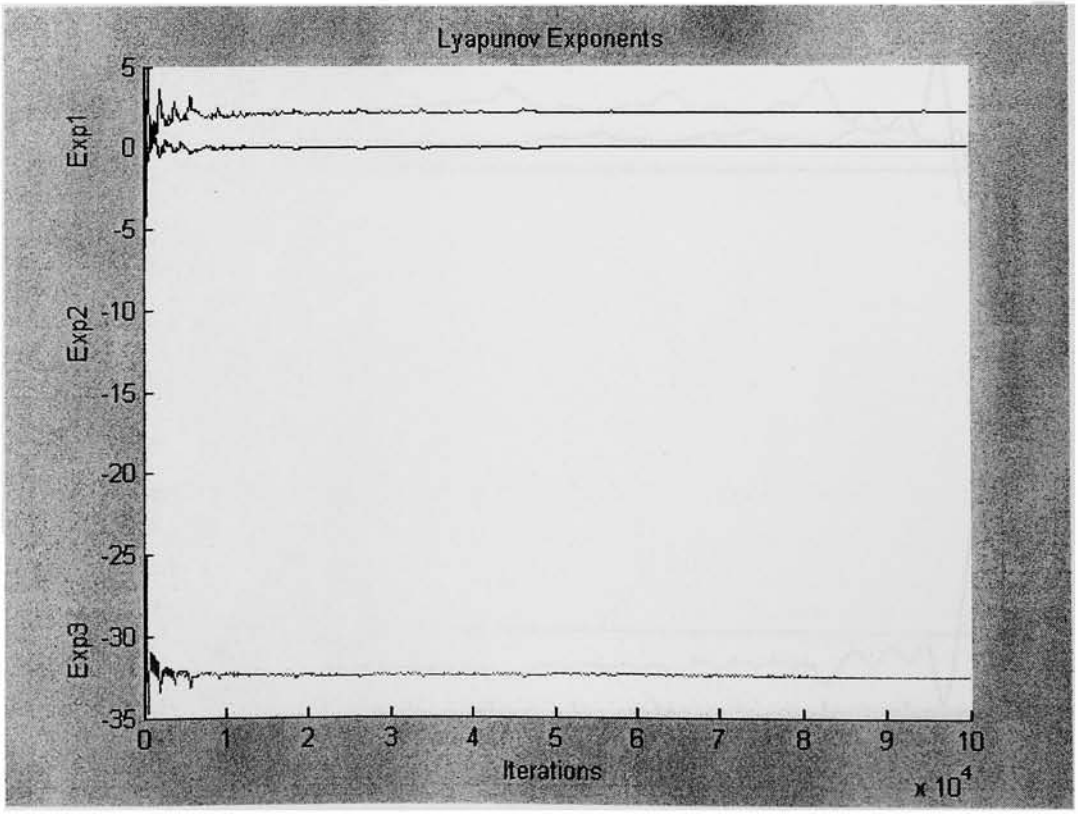
Control Parameters:
System Examined: Lorenz
Parameters: $\sigma = 16$, $r = 45.92$, $b = 4$
Initial Conditions: 1, 0, 0
Time Increment: 0.001
Transient Time: 20
Number of Iterations: 100000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 150.11 seconds

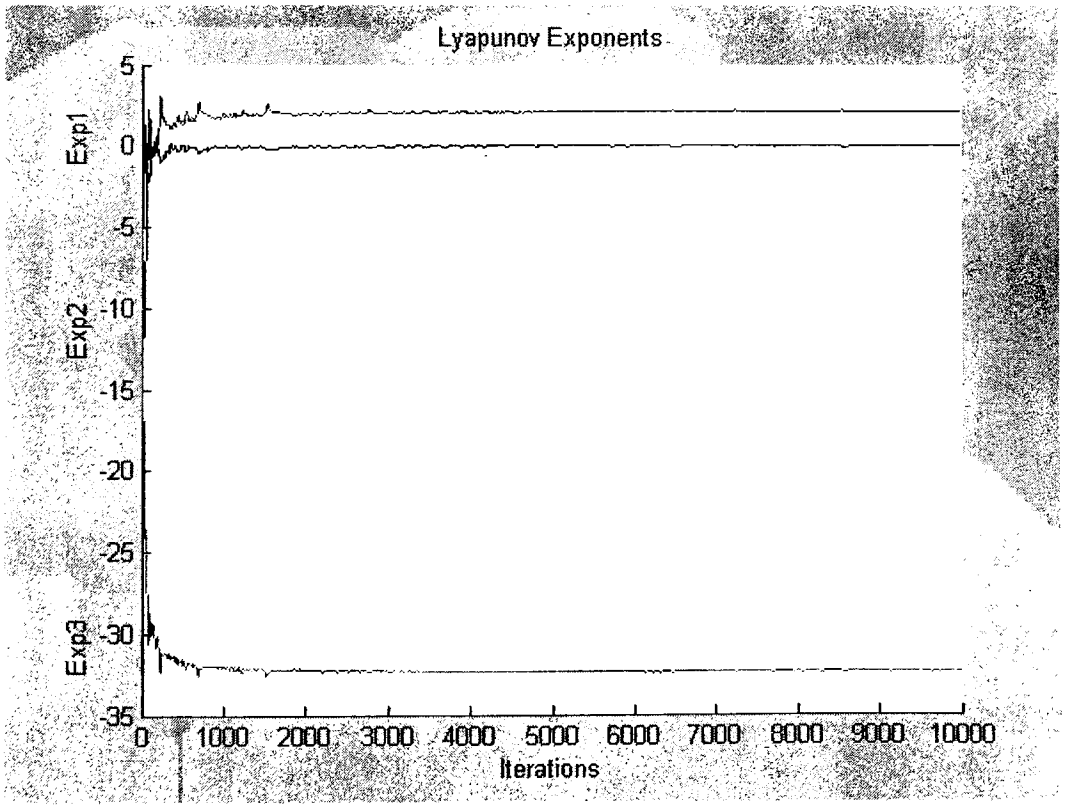
Analysis Results:
Lyapunov Exponent Spectrum: 2.1385, 0.00089854, -32.436
Lyapunov Dimension: 2.066

21:20:40.38
Elapsed time 47 minutes 35.09 seconds
Normal termination of LYAPUNOV_SPECTRUM.M

****



```
**************************************************************
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
19:41:31.07

Control Parameters:
  System Examined: Lorenz
  Parameters: σ = 16, r = 45.92, b = 4
  Initial Conditions: 1, 0, 0
  Time Increment: 0.01
  Transient Time: 20
  Number of Iterations: 10000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 14.99 seconds

Analysis Results:
  Lyapunov Exponent Spectrum: 2.1304, -0.0095797, -32.4174
  Lyapunov Dimension: 2.0654

19:43:30.58
Elapsed time 1 minutes 59.51 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
**************************************************************
```



****

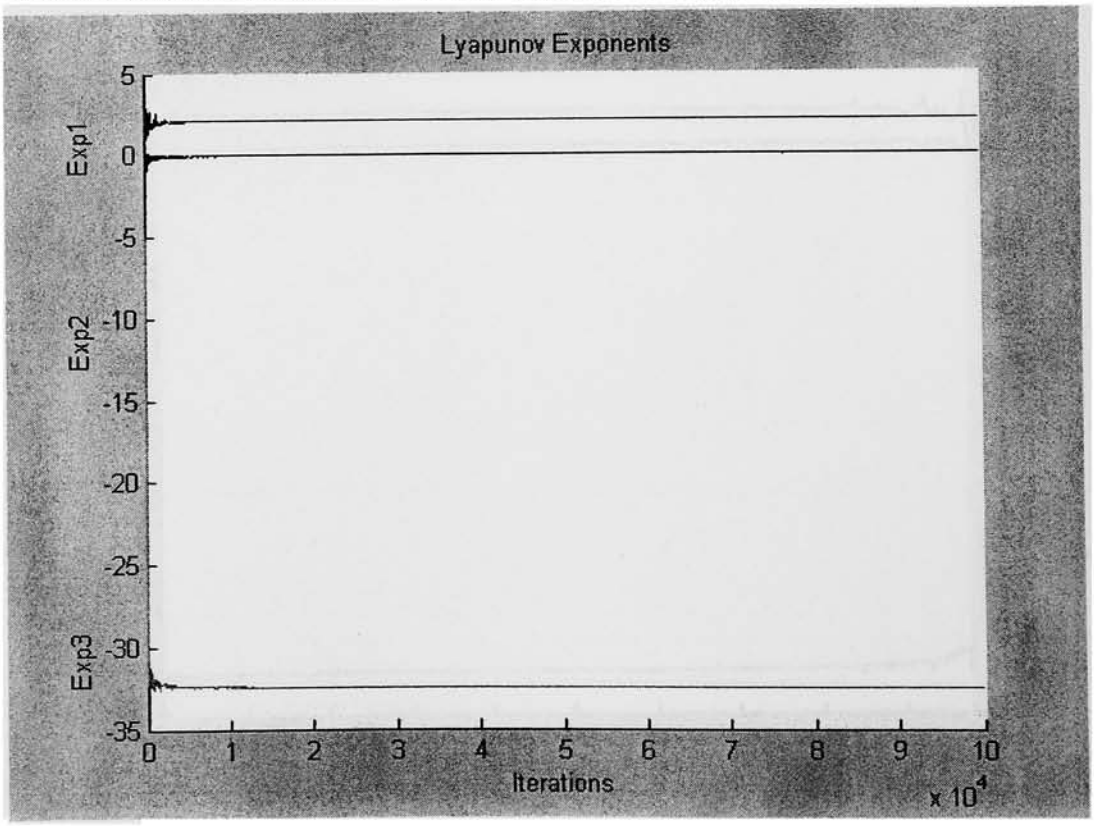
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
19:43:30.58

Control Parameters:
System Examined: Lorenz
Parameters: $\sigma = 16$, $r = 45.92$, $b = 4$
Initial Conditions: 1, 0, 0
Time Increment: 0.01
Transient Time: 20
Number of Iterations: 100000

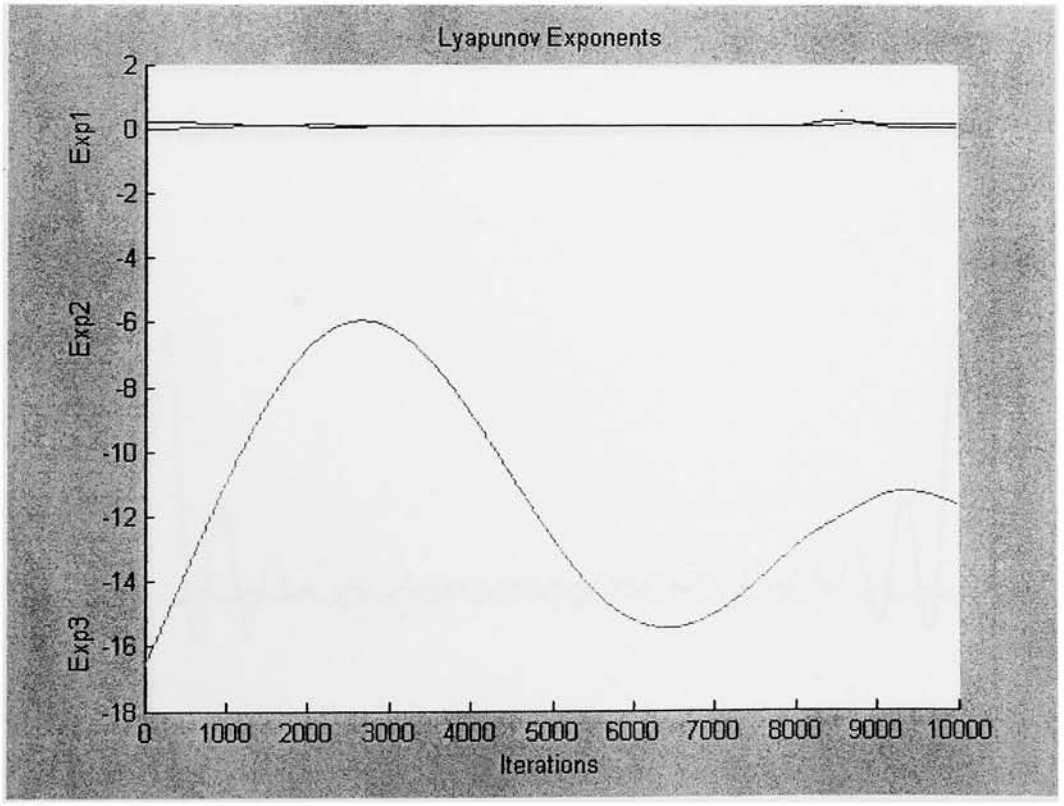
System Iterated So Trajectory will Converge to the Attractor
Elapsed time 15.11 seconds

Analysis Results:
Lyapunov Exponent Spectrum: 2.1727, -0.0011412, -32.4681
Lyapunov Dimension: 2.0669

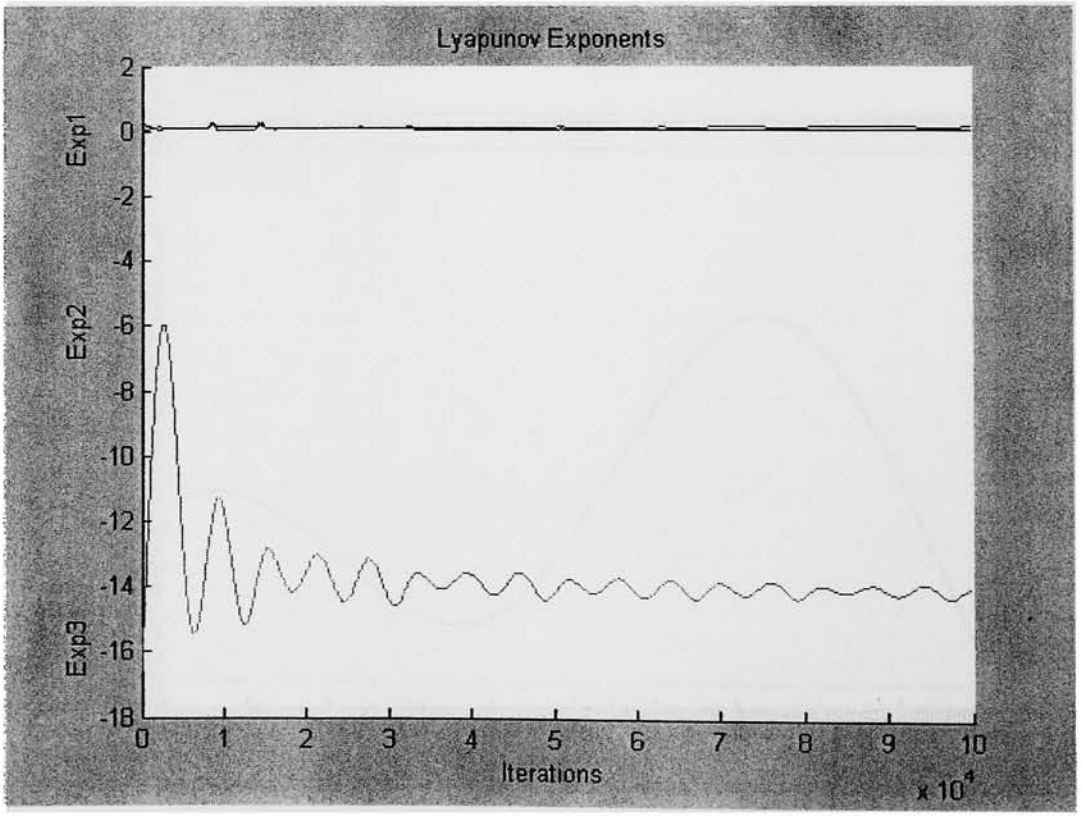
20:28:50.77
Elapsed time 45 minutes 20.19 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
****



```
**********************************************************************
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
21:59:4.23

Control Parameters:
  System Examined: Rössler
  Parameters: a = 0.15, b = 0.20, c = 10
  Initial Conditions: 1, 0, 0
  Time Increment: 0.001
  Transient Time: 60
  Number of Iterations: 10000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 160.1 seconds

Analysis Results:
  Lyapunov Exponent Spectrum: 0.1657, 0.018849, -11.6662
  Lyapunov Dimension: 2.0158

22:2:40.25
Elapsed time 3 minutes 36.02 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
**********************************************************************
```



```
**************************************************************
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
22:2:40.25

Control Parameters:
  System Examined: Rössler
  Parameters: a = 0.15, b = 0.20, c = 10
  Initial Conditions: 1, 0, 0
  Time Increment: 0.001
  Transient Time: 60
  Number of Iterations: 100000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 160.65 seconds

Analysis Results:
  Lyapunov Exponent Spectrum: 0.13571, 0.02459, -13.9724
  Lyapunov Dimension: 2.0115

22:42:12.26
Elapsed time 39 minutes 32.01 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
**************************************************************
```



****

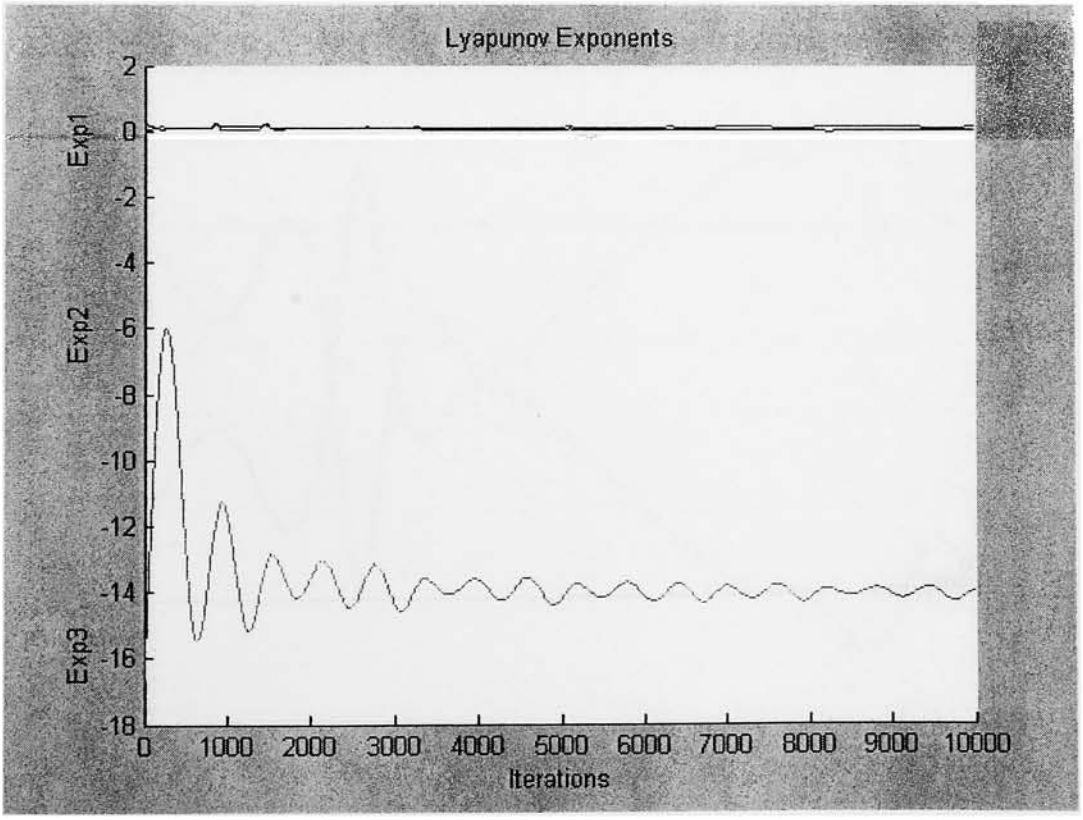
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
21:20:40.38

Control Parameters:
System Examined: Rössler
Parameters: $a = 0.15$, $b = 0.20$, $c = 10$
Initial Conditions: 1, 0, 0
Time Increment: 0.01
Transient Time: 60
Number of Iterations: 10000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 16.14 seconds

Analysis Results:
Lyapunov Exponent Spectrum: 0.13644, 0.024805, -13.9737
Lyapunov Dimension: 2.0115

21:21:51.67
Elapsed time 1 minutes 11.29 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
****



****

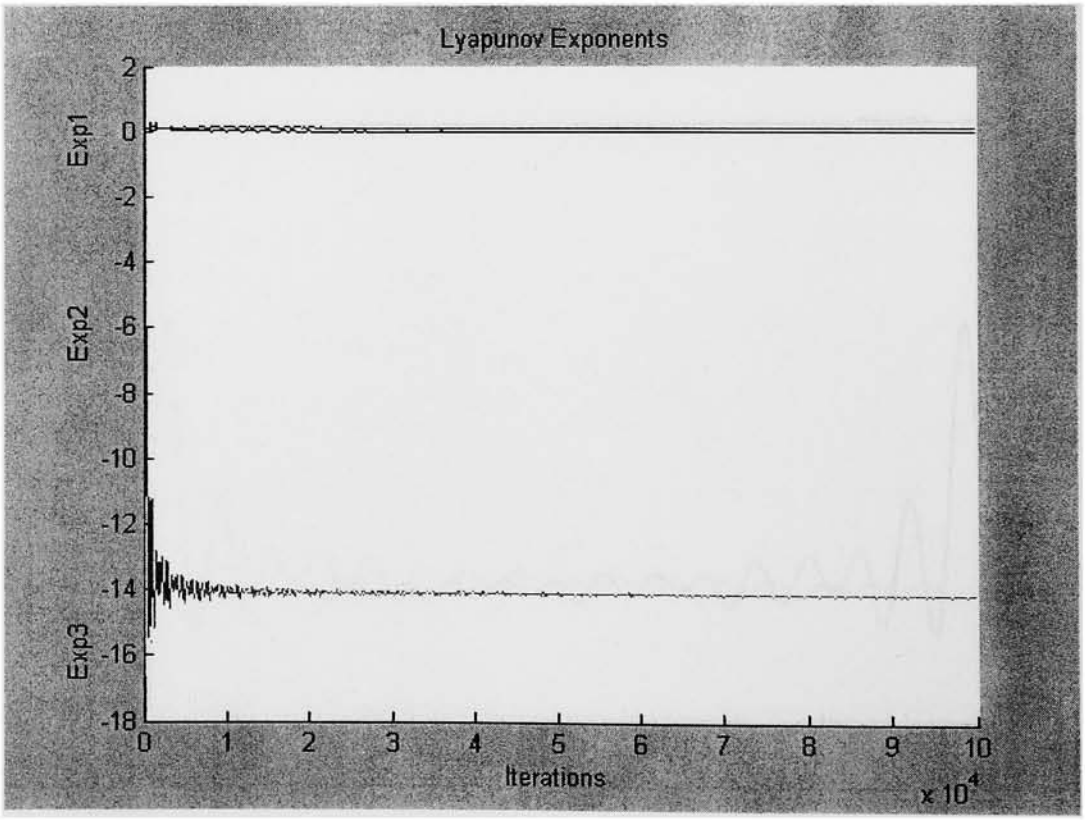
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
21:21:51.67

Control Parameters:
System Examined: Rössler
Parameters: $a = 0.15$, $b = 0.20$, $c = 10$
Initial Conditions: 1, 0, 0
Time Increment: 0.01
Transient Time: 60
Number of Iterations: 100000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 16.04 seconds

Analysis Results:
Lyapunov Exponent Spectrum: 0.1214, 0.0018489, -14.1133
Lyapunov Dimension: 2.0087

21:59:4.23
Elapsed time 37 minutes 12.56 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
****



****

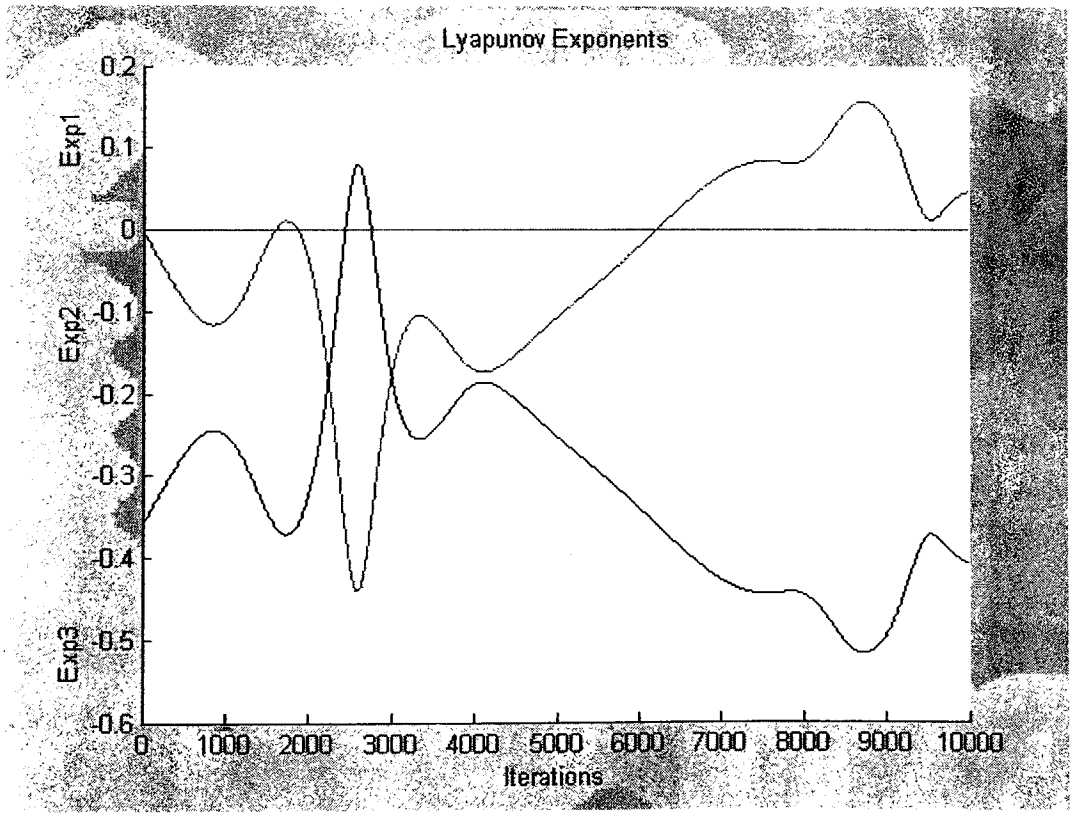
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
23:21:3.35

Control Parameters:
System Examined: Duffing
Parameters: $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Initial Conditions: 1, 0, 0
Time Increment: 0.001
Transient Time: 20
Number of Iterations: 10000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 45.32 seconds

Analysis Results:
Lyapunov Exponent Spectrum: 0.046354, 0, -0.40703
Lyapunov Dimension: 2.1139

23:22:39.09
Elapsed time 1 minutes 35.74 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
****



****

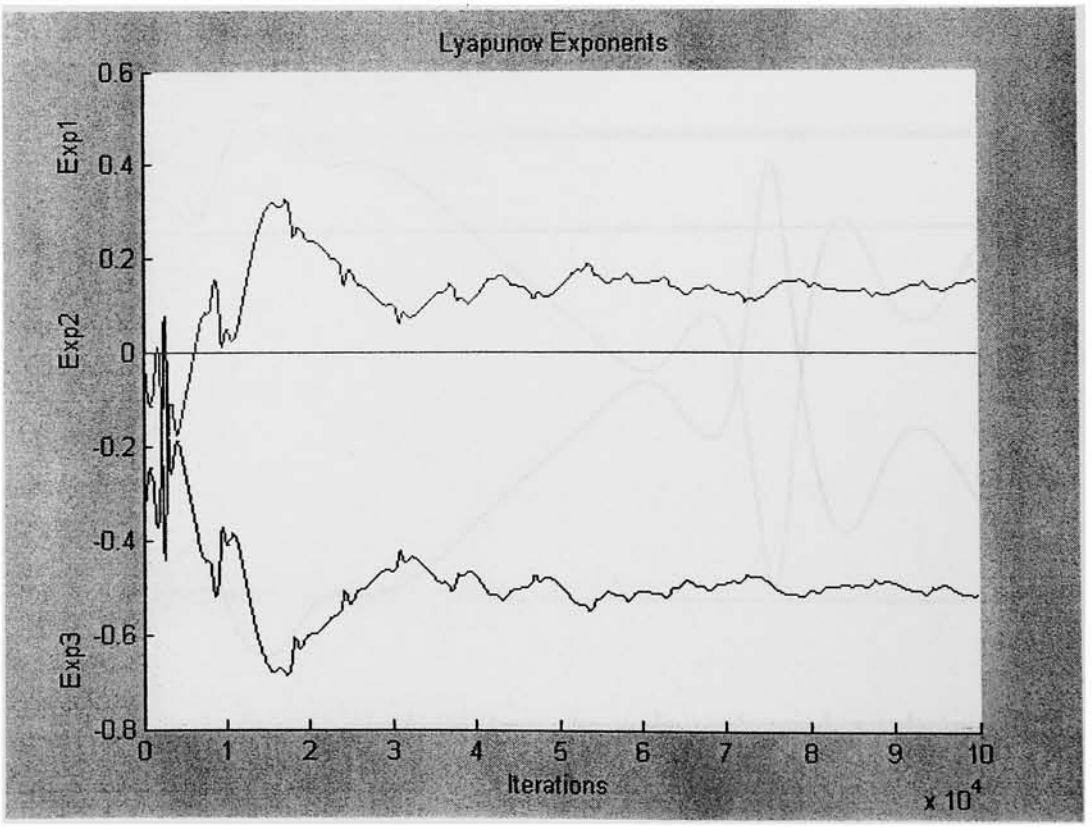
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
23:22:39.09

Control Parameters:
System Examined: Duffing
Parameters: $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Initial Conditions: 1, 0, 0
Time Increment: 0.001
Transient Time: 20
Number of Iterations: 100000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 44.82 seconds

Analysis Results:
Lyapunov Exponent Spectrum: 0.15251, 0, -0.51318
Lyapunov Dimension: 2.2972

0:1:5.58
Elapsed time 38 minutes 26.49 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
****



****

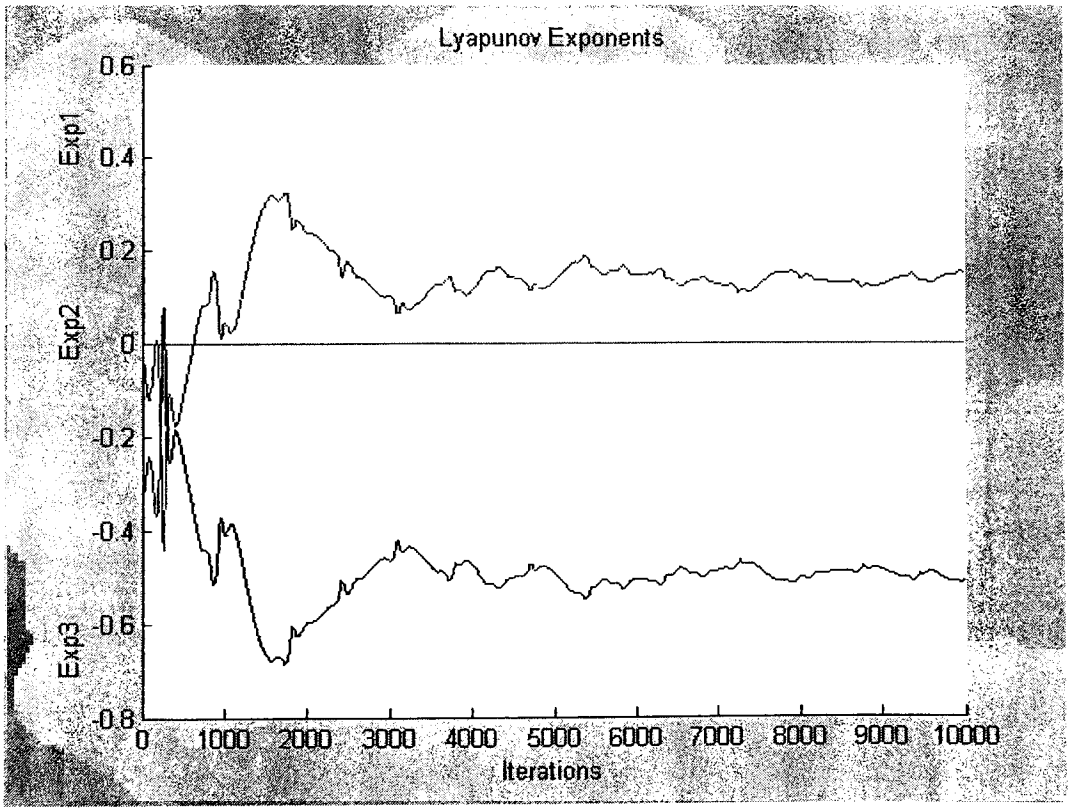
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
22:42:12.26

Control Parameters:
System Examined: Duffing
Parameters: $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Initial Conditions: 1, 0, 0
Time Increment: 0.01
Transient Time: 20
Number of Iterations: 10000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 4.67 seconds

Analysis Results:
Lyapunov Exponent Spectrum: 0.15249, 0, -0.51316
Lyapunov Dimension: 2.2972

22:43:6.86
Elapsed time 0 minutes 54.6 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
****



****

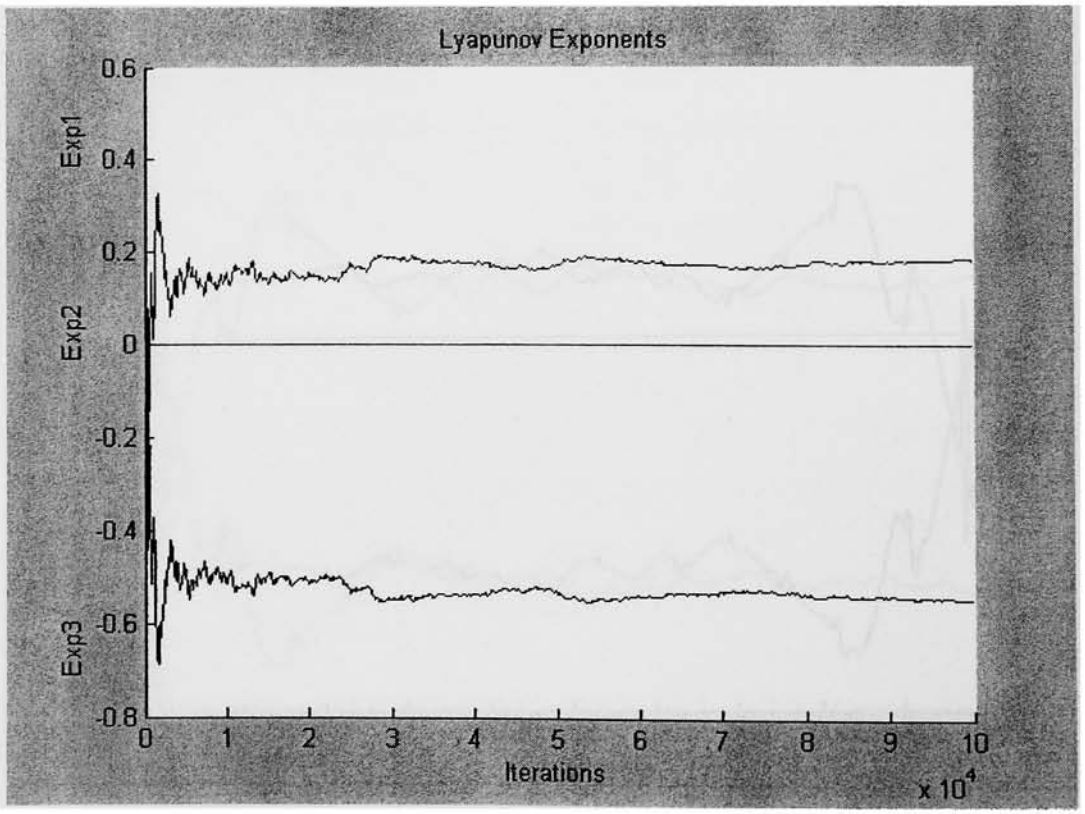
Lyapunov Spectrum calculation program by Andrew Dick
25-Jul-2003
22:43:6.86

Control Parameters:
System Examined: Duffing
Parameters: $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Initial Conditions: 1, 0, 0
Time Increment: 0.01
Transient Time: 20
Number of Iterations: 100000

System Iterated So Trajectory will Converge to the Attractor
Elapsed time 4.55 seconds

Analysis Results:
Lyapunov Exponent Spectrum: 0.1822, 0, -0.54288
Lyapunov Dimension: 2.3356

23:21:3.35
Elapsed time 37 minutes 56.49 seconds
Normal termination of LYAPUNOV_SPECTRUM.M
****



APPENDIX C2

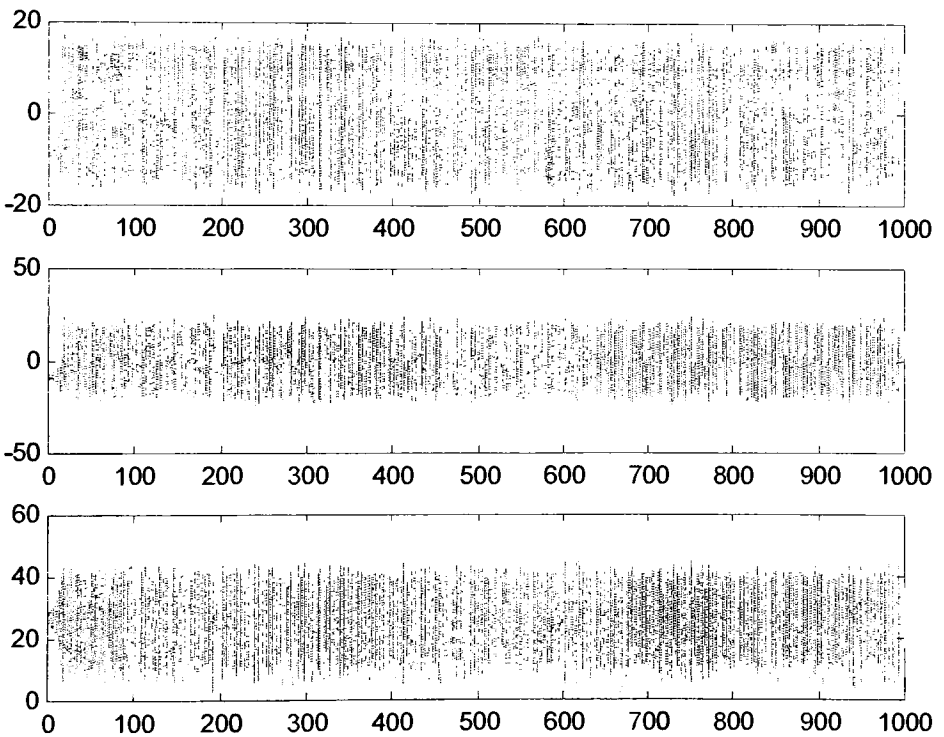
****

Lie Series approximation iteration program by Andrew Dick
26-Jul-2003
9:1:6.953

Iteration Parameters
5th order approximation of Lorenz system
Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Time step of 0.01 seconds
Magnitude of noise added: 0.005
Length of iteration 1000 seconds
Initial conditions: 1,0,0

9:7:16.654
Elapsed time 6 minutes 9.711 seconds
Normal Termination of LIE_SERIES_ITERATION.M

****

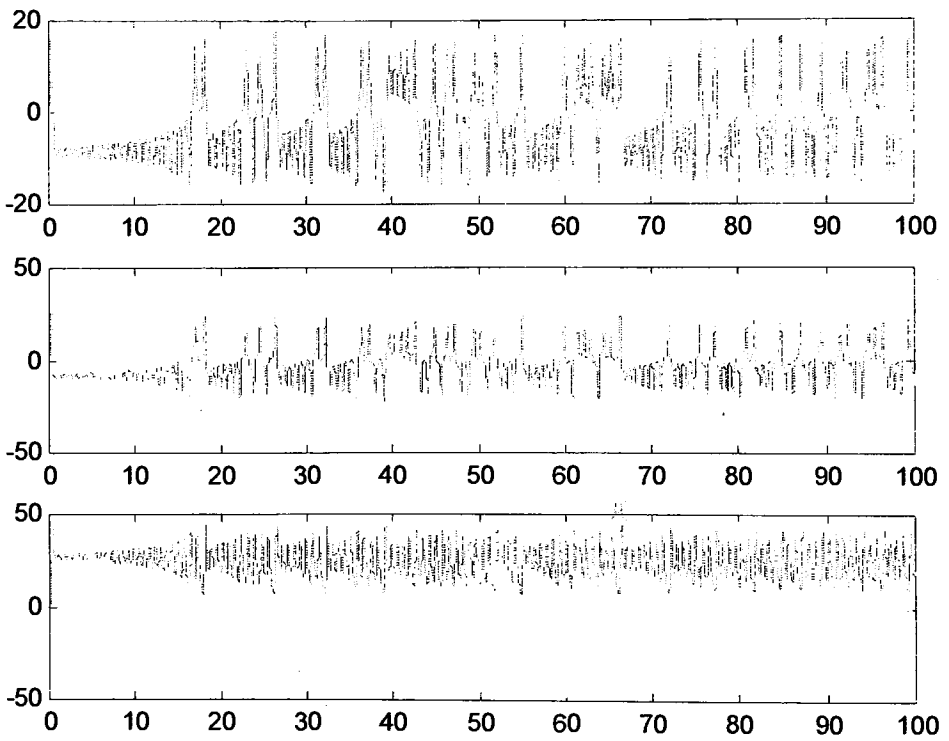


****

Lie Series approximation iteration program by Andrew Dick
26-Jul-2003
9:7:16.834

Iteration Parameters
5th order approximation of Lorenz system
Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Time step of 0.001 seconds
Magnitude of noise added: 0.005
Length of iteration 100 seconds
Initial conditions: 1,0,0

9:13:54.657
Elapsed time 6 minutes 37.823 seconds
Normal Termination of LIE_SERIES_ITERATION.M
****

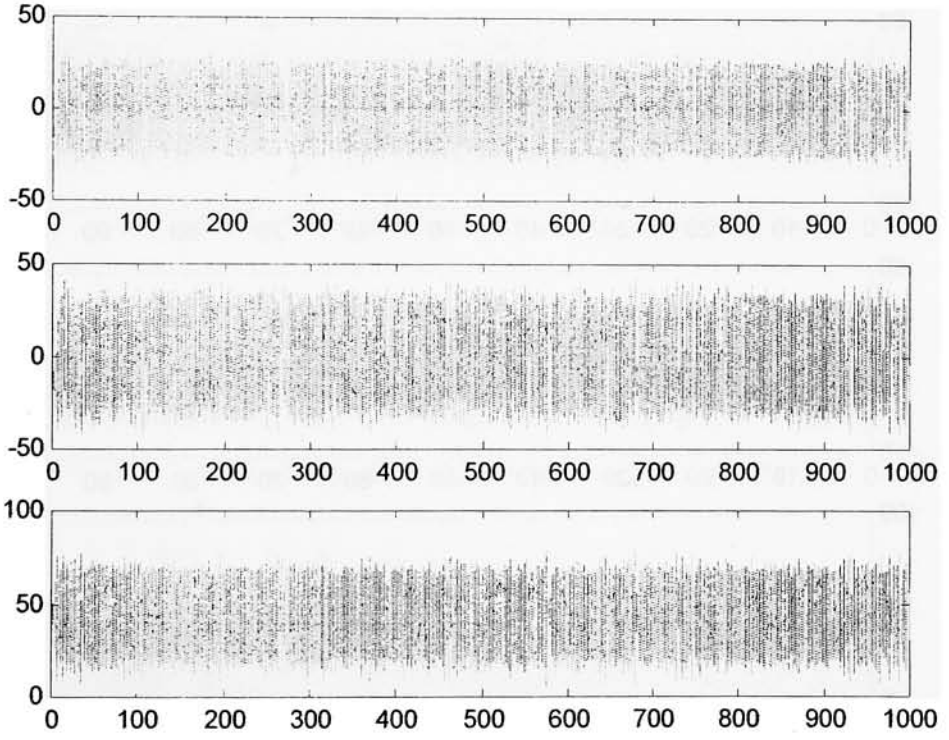


****

Lie Series approximation iteration program by Andrew Dick
26-Jul-2003
9:13:54.727

Iteration Parameters
5th order approximation of Lorenz system
Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Time step of 0.01 seconds
Magnitude of noise added: 0.005
Length of iteration 1000 seconds
Initial conditions: 1,0,0

9:20:20.371
Elapsed time 6 minutes 25.644 seconds
Normal Termination of LIE_SERIES_ITERATION.M
****

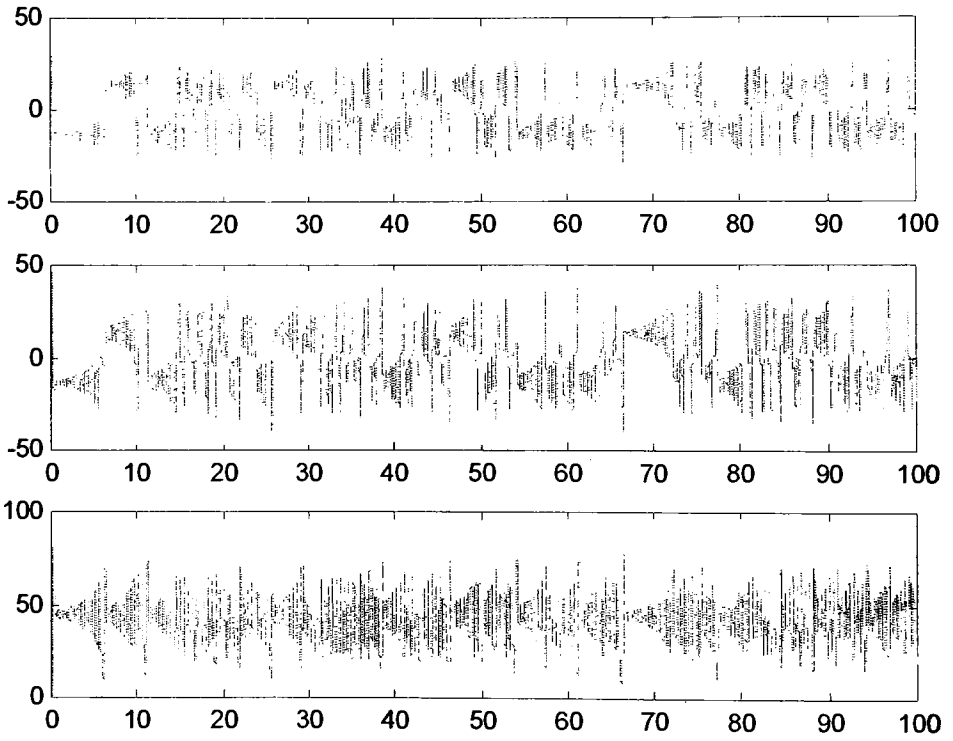


****

Lie Series approximation iteration program by Andrew Dick
26-Jul-2003
9:20:20.491

Iteration Parameters
5th order approximation of Lorenz system
Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Time step of 0.001 seconds
Magnitude of noise added: 0.005
Length of iteration 100 seconds
Initial conditions: 1,0,0

9:27:50.478
Elapsed time 7 minutes 29.987 seconds
Normal Termination of LIE_SERIES_ITERATION.M
****



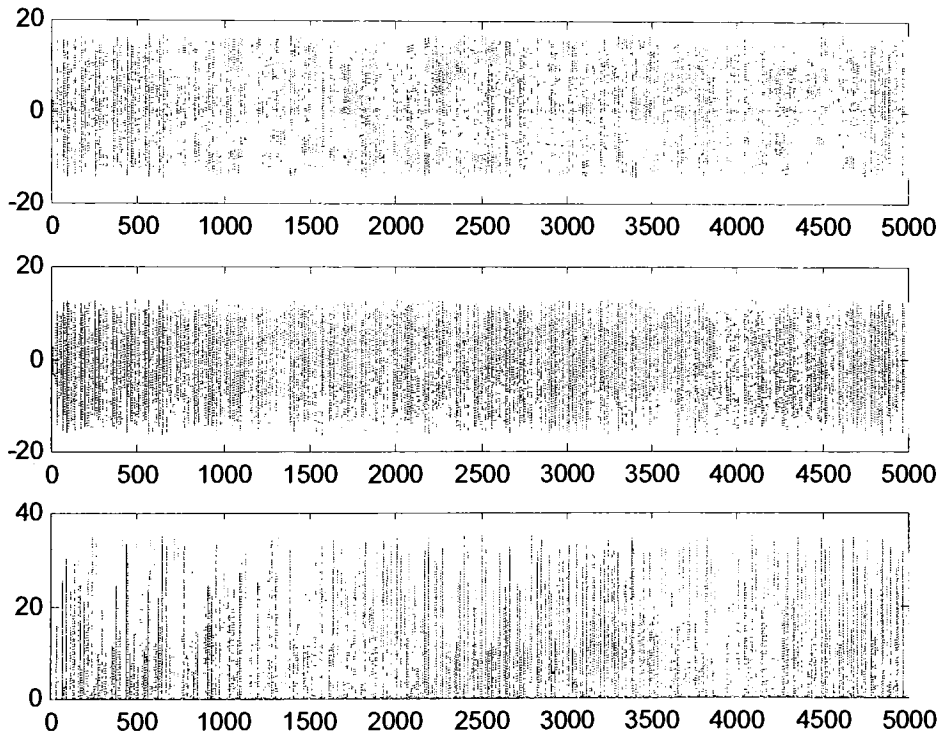
****

Lie Series approximation iteration program by Andrew Dick
28-Jul-2003
17:56:35.524

Iteration Parameters
5th order approximation of Rössler system
Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Time step of 0.05 seconds
Magnitude of noise added: 0.005
Length of iteration 5000 seconds
Initial conditions: 1,0,0

17:58:58.54
Elapsed time 2 minutes 23.016 seconds
Normal Termination of LIE_SERIES_ITERATION.M

****

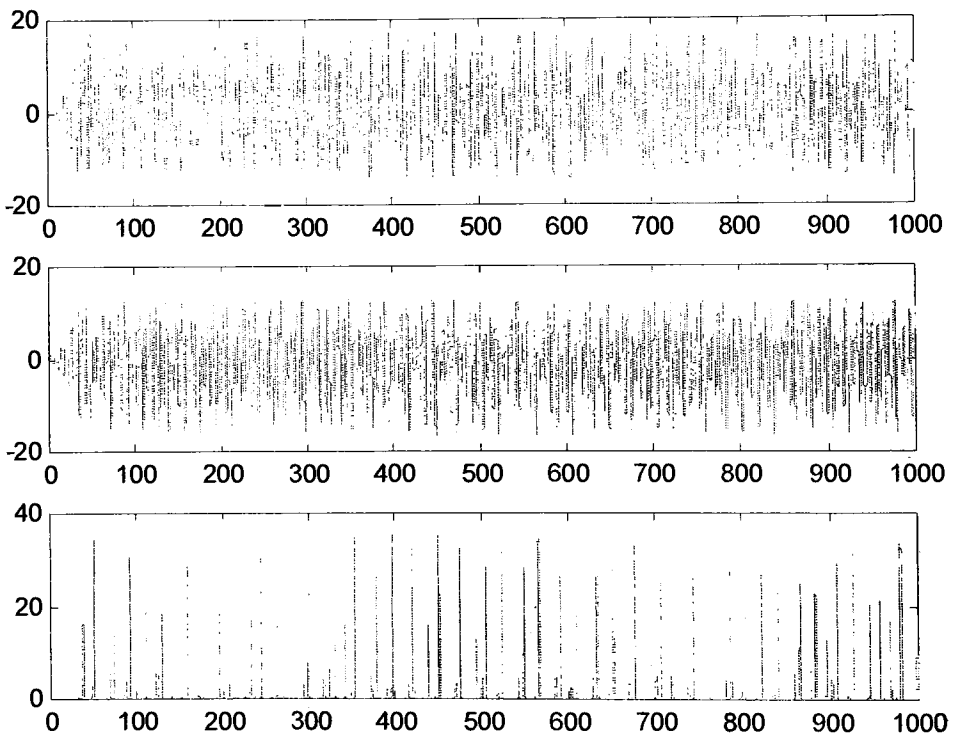


****

Lie Series approximation iteration program by Andrew Dick
26-Jul-2003
9:28:42.804

Iteration Parameters
5th order approximation of Rössler system
Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Time step of 0.01 seconds
Magnitude of noise added: 0.005
Length of iteration 1000 seconds
Initial conditions: 1,0,0

9:30:52.17
Elapsed time 2 minutes 9.366 seconds
Normal Termination of LIE_SERIES_ITERATION.M
****

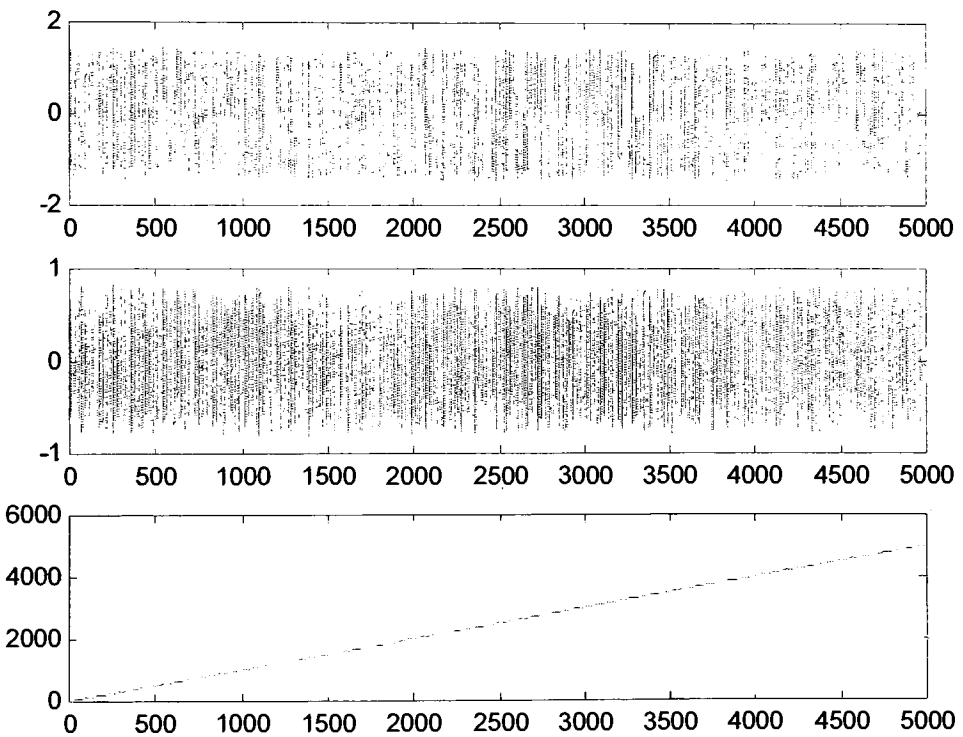


****

Lie Series approximation iteration program by Andrew Dick
28-Jul-2003
17:58:58.61

Iteration Parameters
5th order approximation of Duffing system
Popular Parameter Set, $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Time step of 0.05 seconds
Magnitude of noise added: 0.001
Length of iteration 5000 seconds
Initial conditions: 1,0,0

18:1:18.031
Elapsed time 2 minutes 19.421 seconds
Normal Termination of LIE_SERIES_ITERATION.M
****

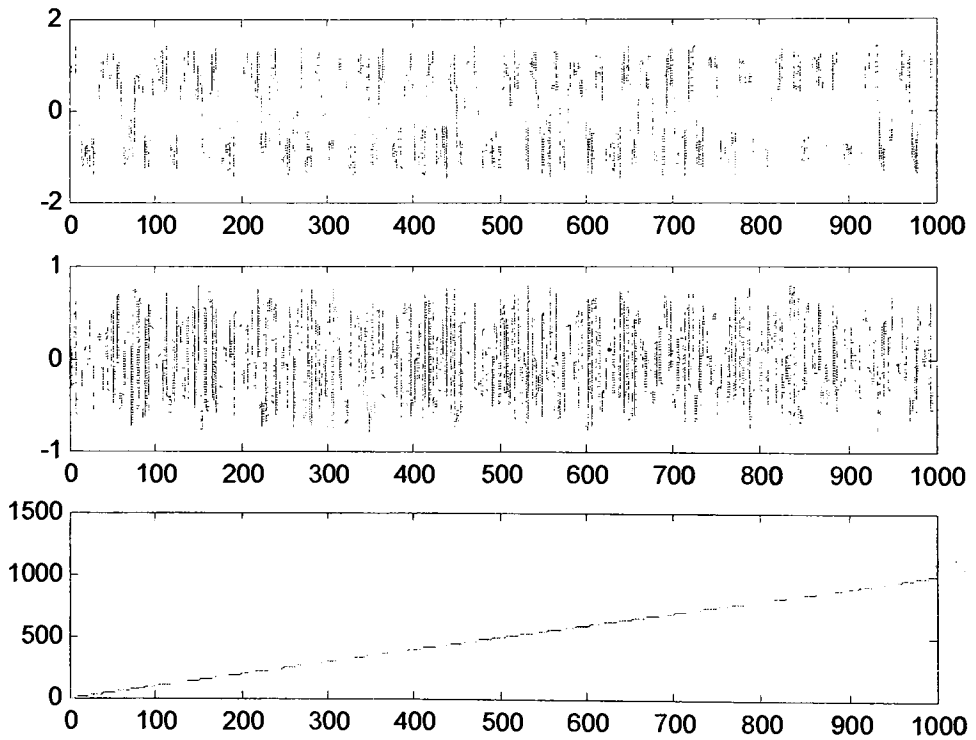


****

Lie Series approximation iteration program by Andrew Dick
26-Jul-2003
9:31:36.784

Iteration Parameters
5th order approximation of Duffing system
Popular Parameter Set, $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Time step of 0.01 seconds
Magnitude of noise added: 0.001
Length of iteration 1000 seconds
Initial conditions: 1,0,0

9:33:27.213
Elapsed time 1 minutes 50.429 seconds
Normal Termination of LIE_SERIES_ITERATION.M
****



APPENDIX C3

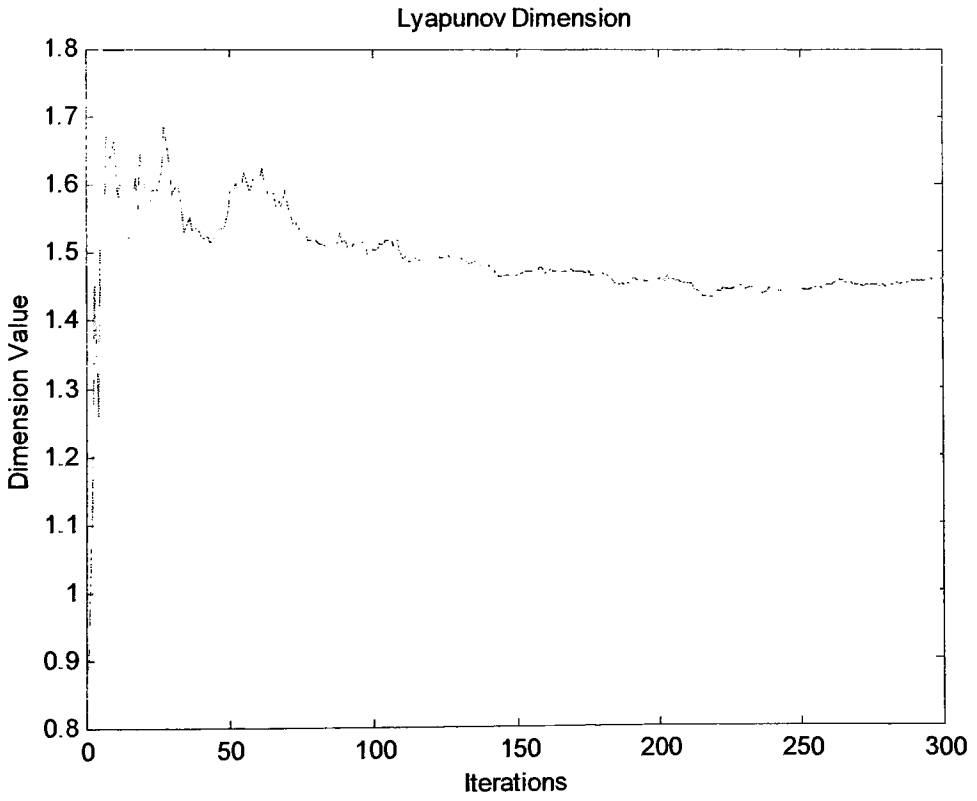
****

Largest Lyapunov exponent calculation program by Andrew Dick
Data previously iterated using 5th order Lie Series approximation
First 20 seconds of transient trajectory removed
29-Jul-2003
23:4:7.443

5th order approximation of Lorenz system
Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Calculated from 90000 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 1.4598

2:13:25.736
Elapsed time 189 minutes 18.583 seconds
Normal Termination of LARGEST_LYAPUNOV.M

****

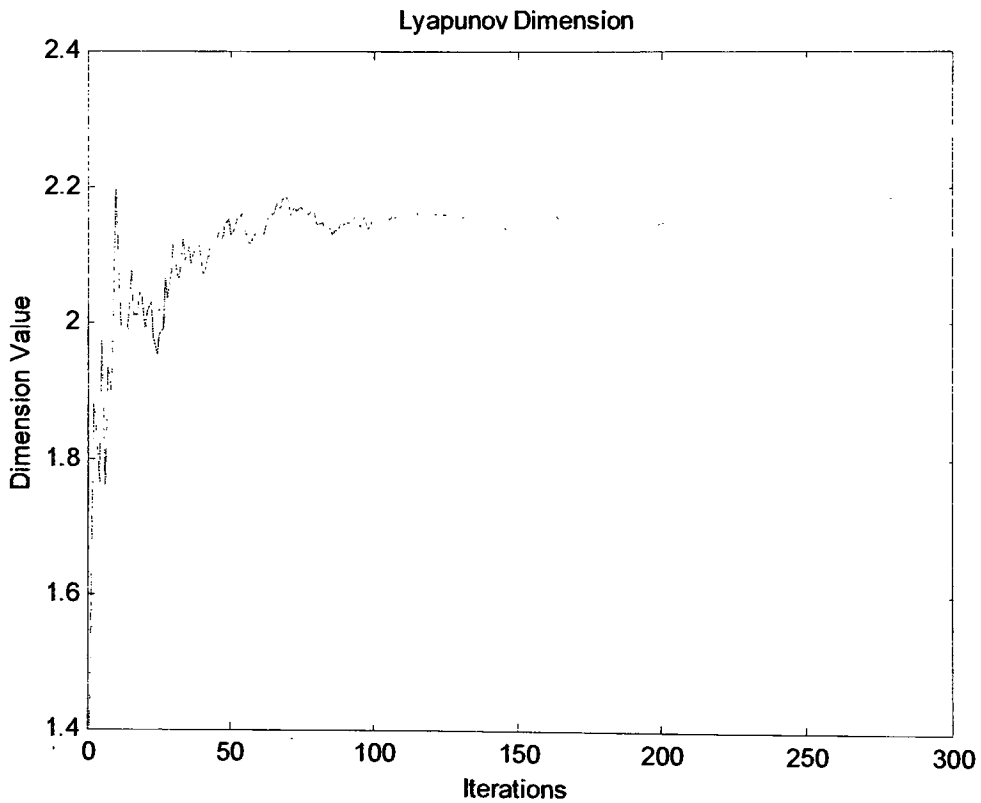


****

Largest Lyapunov exponent calculation program by Andrew Dick
Data previously iterated using 5th order Lie Series approximation
First 10 seconds of transient trajectory removed
30-Jul-2003
2:13:26.407

5th order approximation of Lorenz system
Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Calculated from 90000 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 2.1763

5:9:40.181
Elapsed time 176 minutes 13.774 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



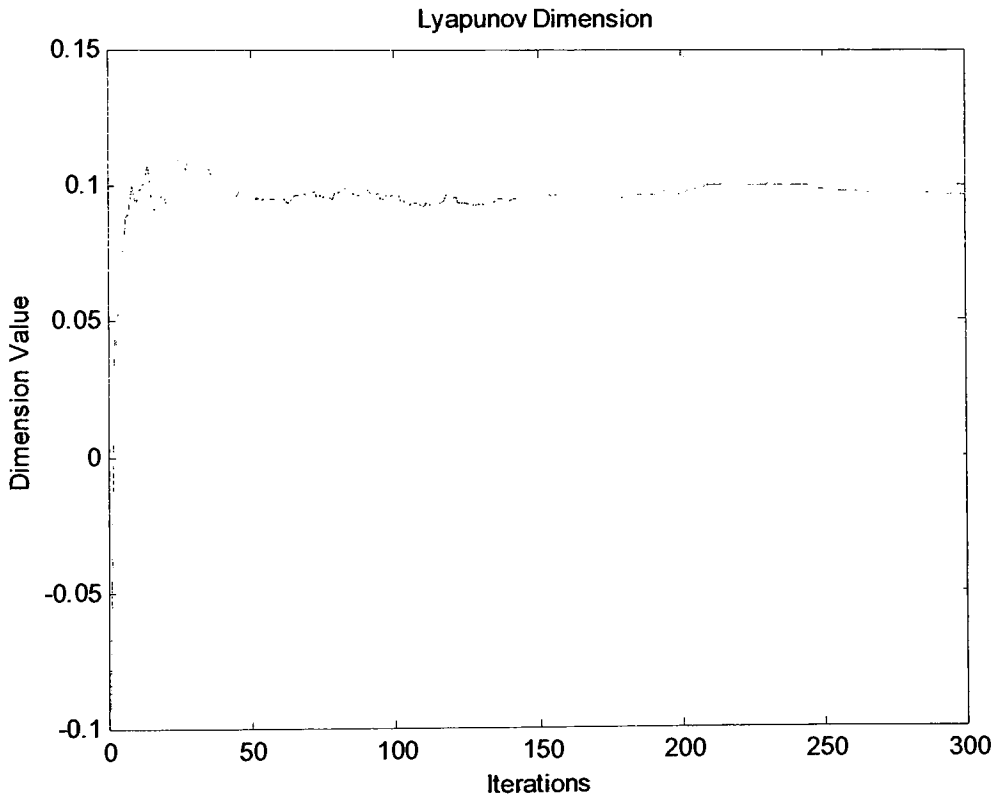
****

Largest Lyapunov exponent calculation program by Andrew Dick
Data previously iterated using 5th order Lie Series approximation
First 50 seconds of transient trajectory removed
30-Jul-2003
5:9:40.432

5th order approximation of Rössler system
Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Calculated from 90000 data points with time step of 0.05 seconds
Calculated using 300 12.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.096446

8:11:35.987
Elapsed time 181 minutes 55.555 seconds
Normal Termination of LARGEST_LYAPUNOV.M

****

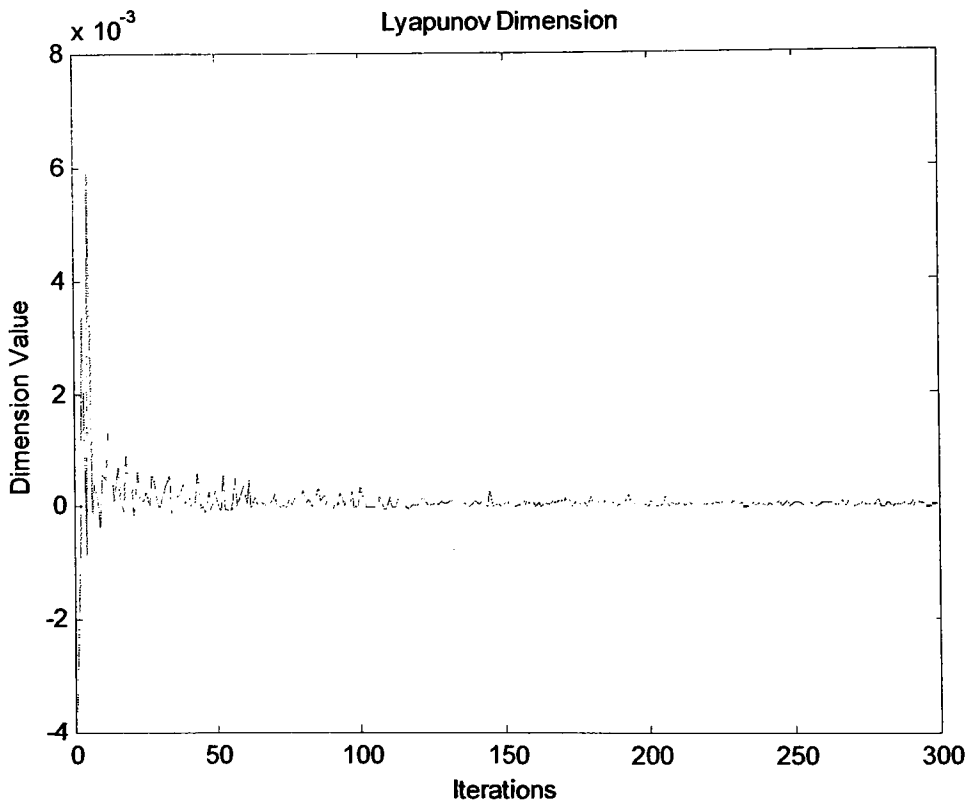


****

Largest Lyapunov exponent calculation program by Andrew Dick
Data previously iterated using 5th order Lie Series approximation
First 20 seconds of transient trajectory removed
30-Jul-2003
8:11:36.498

5th order approximation of Duffing system
Popular Parameter Set, $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Calculated from 90000 data points with time step of 0.05 seconds
Calculated using 300 12.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 8.1145e-006

11:37:56.149
Elapsed time 206 minutes 19.651 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



APPENDIX C4

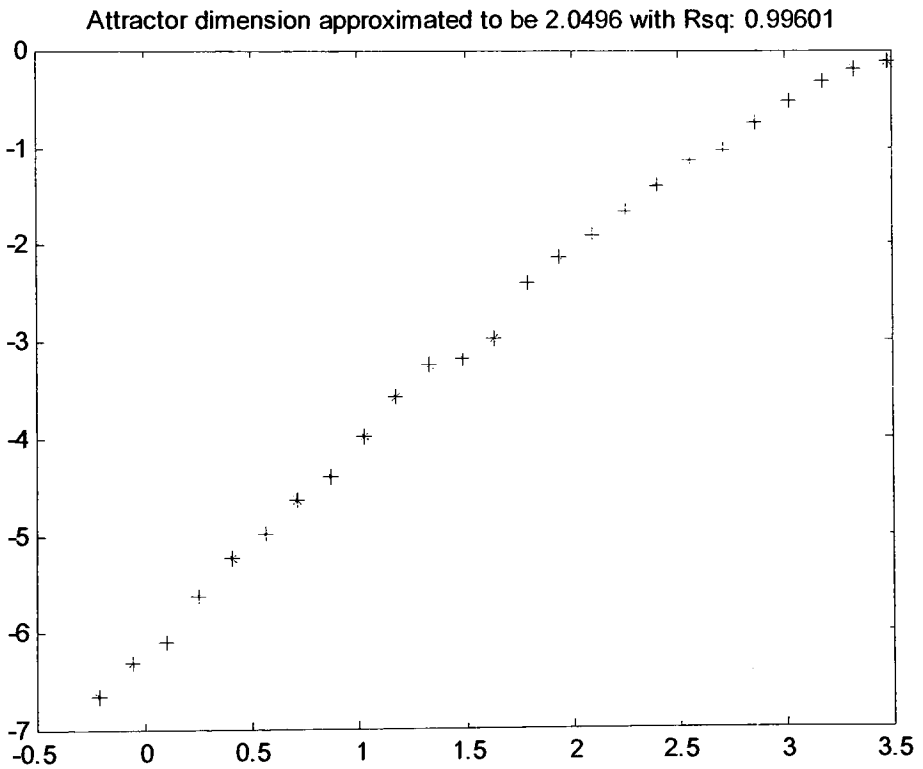
****

Correlation Dimension calculation program by Andrew Dick
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
First 20 seconds of transient trajectory removed
Calculated using 5000 data points and examining 250 random points
Calculated using a Theiler coefficient of W=10
12-Aug-2003
14:54:30.29

The Correlation Dimension of the data set was determined to be 2.0496

14:58:49.222
Elapsed time 4 minutes 18.932 seconds
Normal termination of CORRELATION_DIMENSION.M

****



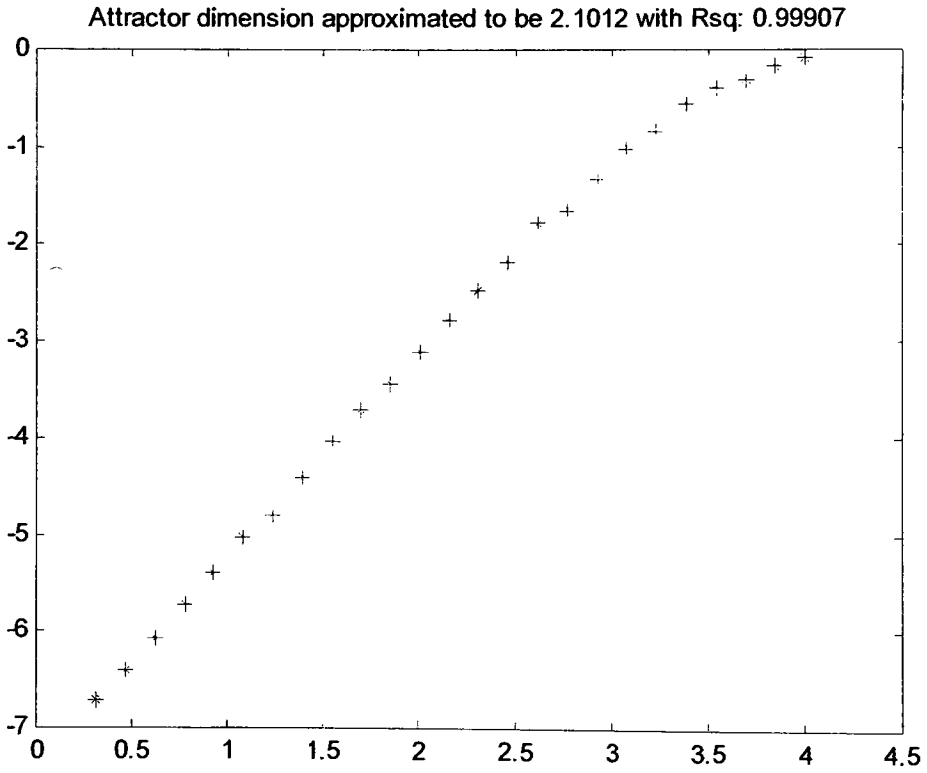
****

Correlation Dimension calculation program by Andrew Dick
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
First 10 seconds of transient trajectory removed
Calculated using 5000 data points and examining 250 random points
Calculated using a Theiler coefficient of W=10
12-Aug-2003
14:58:49.402

The Correlation Dimension of the data set was determined to be 2.1012

15:2:42.387
Elapsed time 3 minutes 52.985 seconds
Normal termination of CORRELATION_DIMENSION.M

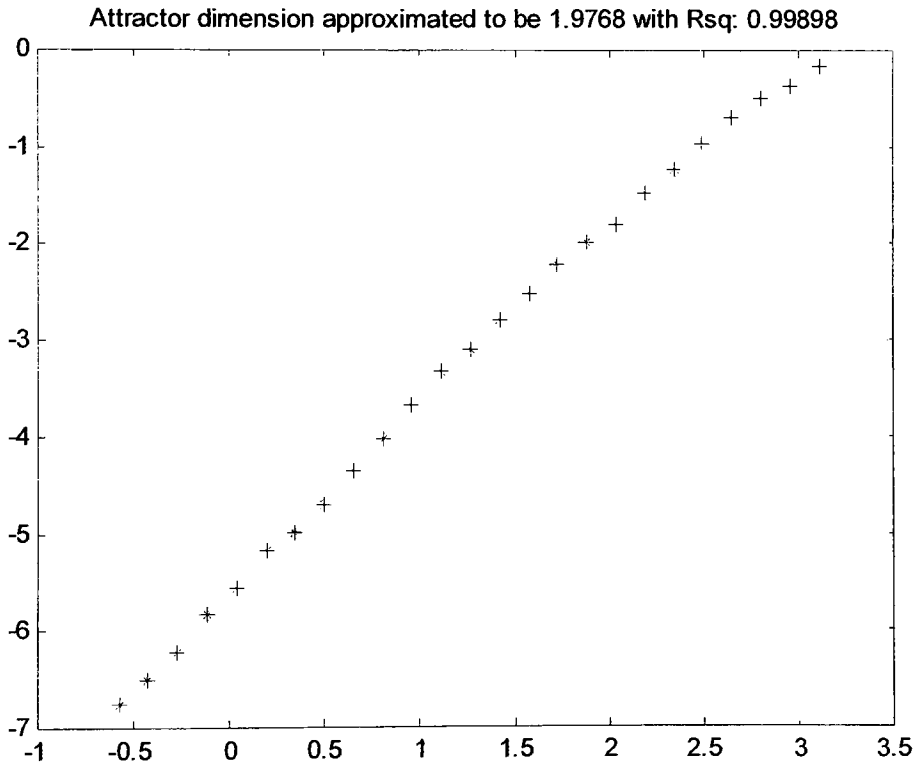
****



****
Correlation Dimension calculation program by Andrew Dick
Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
First 40 seconds of transient trajectory removed
Calculated using 5000 data points and examining 250 random points
Calculated using a Theiler coefficient of W=10
12-Aug-2003
15:2:42.607

The Correlation Dimension of the data set was determined to be 1.9768

15:6:37.185
Elapsed time 3 minutes 54.588 seconds
Normal termination of CORRELATION_DIMENSION.M
****



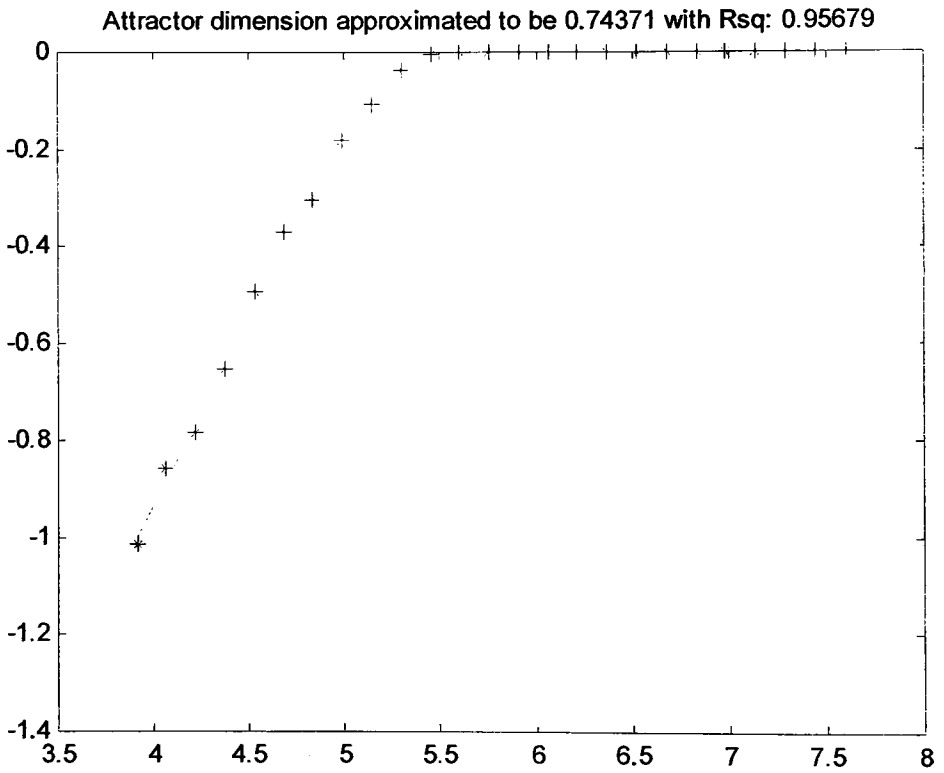
****

Correlation Dimension calculation program by Andrew Dick
Data previously iterated using 5th order Lie Series approximation
Duffing system with Popular Parameter Set, $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
First 20 seconds of transient trajectory removed
Calculated using 5000 data points and examining 250 random points
Calculated using a Theiler coefficient of W=10
12-Aug-2003
15:6:37.255

The Correlation Dimension of the data set was determined to be 0.74371

15:12:2.863
Elapsed time 5 minutes 25.608 seconds
Normal termination of CORRELATION_DIMENSION.M

****



APPENDIX C5

****

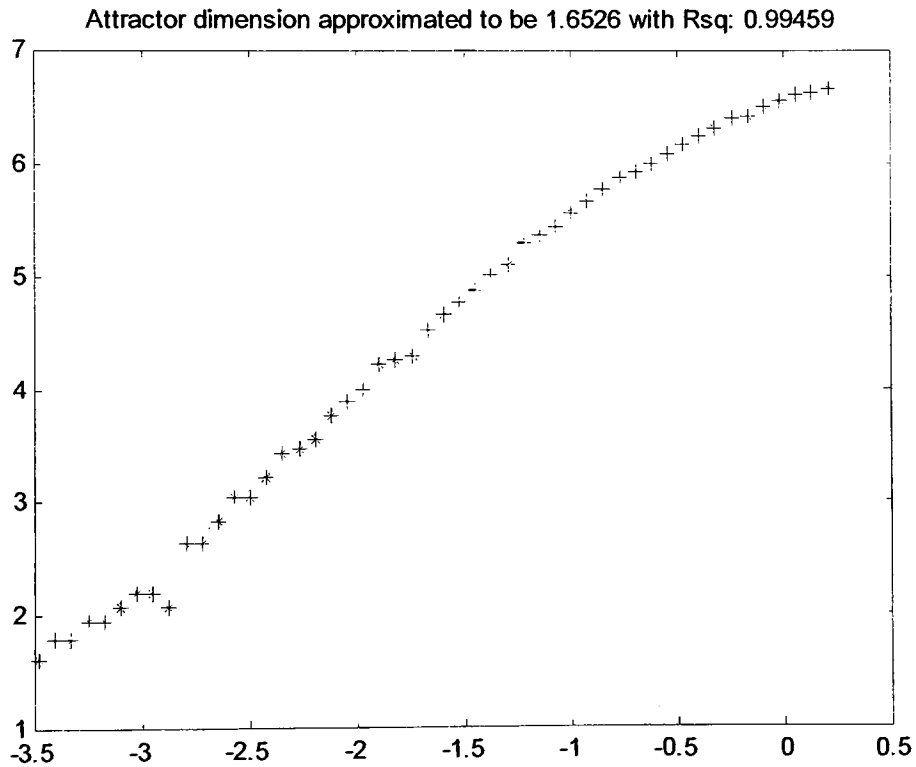
Program to calculate Capacity Dimension by Andrew Dick
11-Aug-2003
20:7:14.774

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Data set consisted of 1000 data points with a time step of 0.01 seconds
A transient period of 20 seconds was removed

The Capacity Dimension of the data set was determined to be 1.6526

20:47:10.659
Elapsed time 39 minutes, 55.885 seconds
Normal Termination of Program CAPACITY_DIMENSION.M

****



****

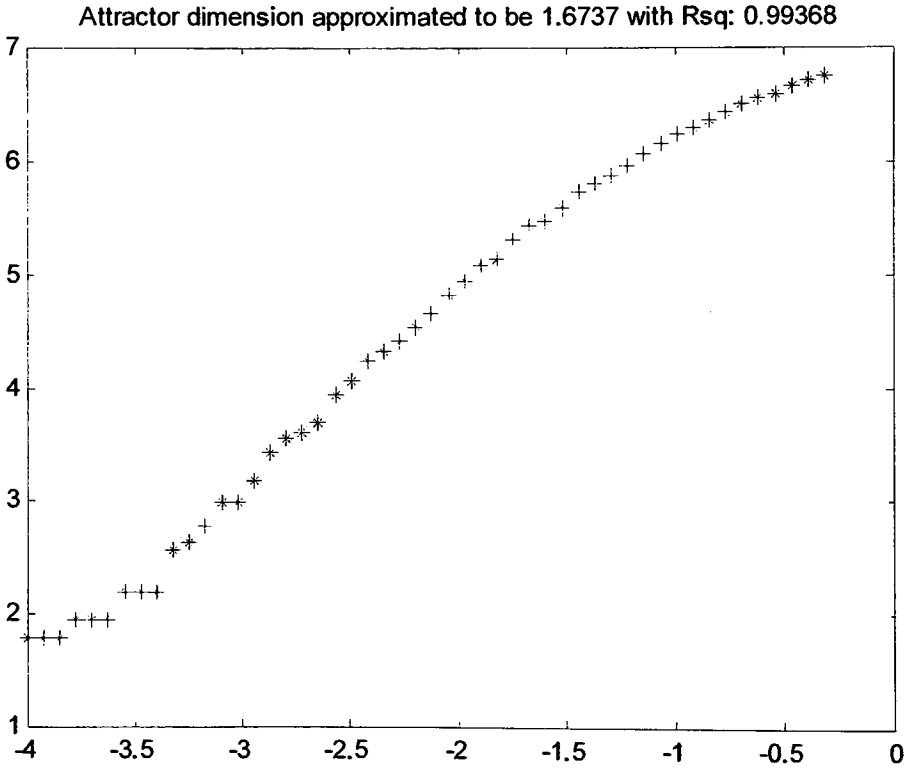
Program to calculate Capacity Dimension by Andrew Dick
11-Aug-2003
20:47:10.679

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Data set consisted of 1000 data points with a time step of 0.01 seconds
A transient period of 50 seconds was removed

The Capacity Dimension of the data set was determined to be 1.6737

21:24:2.049
Elapsed time 36 minutes, 51.37 seconds
Normal Termination of Program CAPACITY_DIMENSION.M

****



****

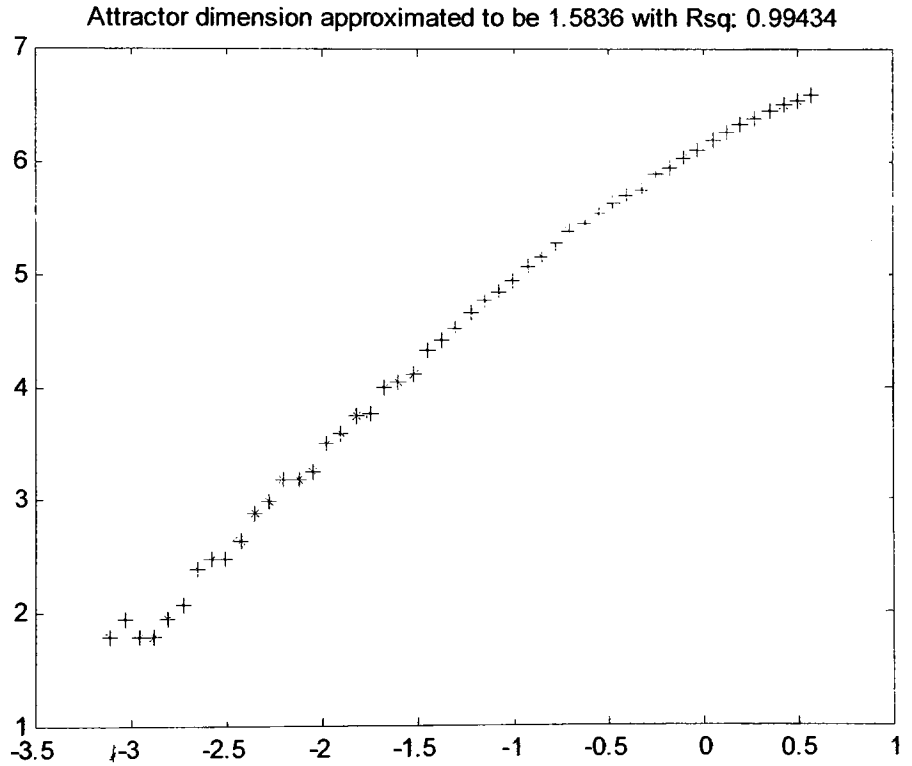
Program to calculate Capacity Dimension by Andrew Dick
11-Aug-2003
21:24:2.059

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Data set consisted of 1000 data points with a time step of 0.05 seconds
A transient period of 40 seconds was removed

The Capacity Dimension of the data set was determined to be 1.5836

22:39:5.034
Elapsed time 75 minutes, 2.975 seconds
Normal Termination of Program CAPACITY_DIMENSION.M

****



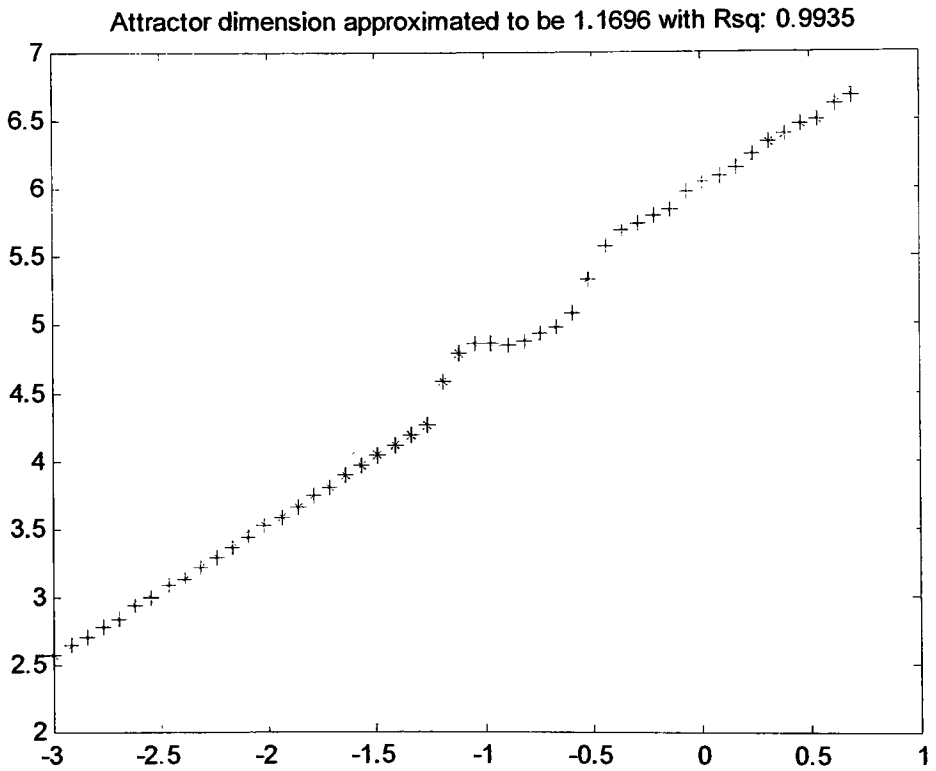
****

Program to calculate Capacity Dimension by Andrew Dick
13-Aug-2003
13:18:41.284

Data previously iterated using 5th order Lie Series approximation
Duffing system with Popular Parameter Set, $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Data set consisted of 5000 data points with a time step of 0.05 seconds
A transient period of 20 seconds was removed

The Capacity Dimension of the data set was determined to be 1.1696

14:4:4.139
Elapsed time 45 minutes, 22.925 seconds
Normal Termination of Program CAPACITY_DIMENSION.M
****



APPENDIX C6

****

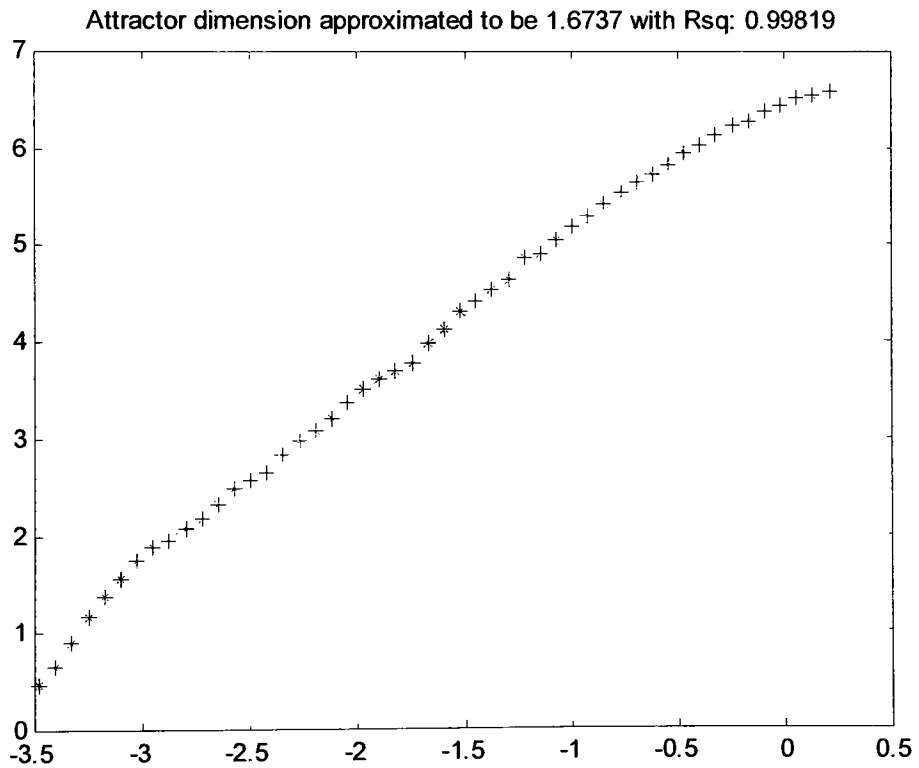
Program to calculate Information Dimension by Andrew Dick
11-Aug-2003
22:39:5.054

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Data set consisted of 1000 data points with a time step of 0.01 seconds
A transient period of 20 seconds was removed

The Information Dimension of the data set was determined to be 1.6737

23:19:0.899
Elapsed time 39 minutes, 55.845 seconds
Normal Termination of Program INFORMATION_DIMENSION.M

****



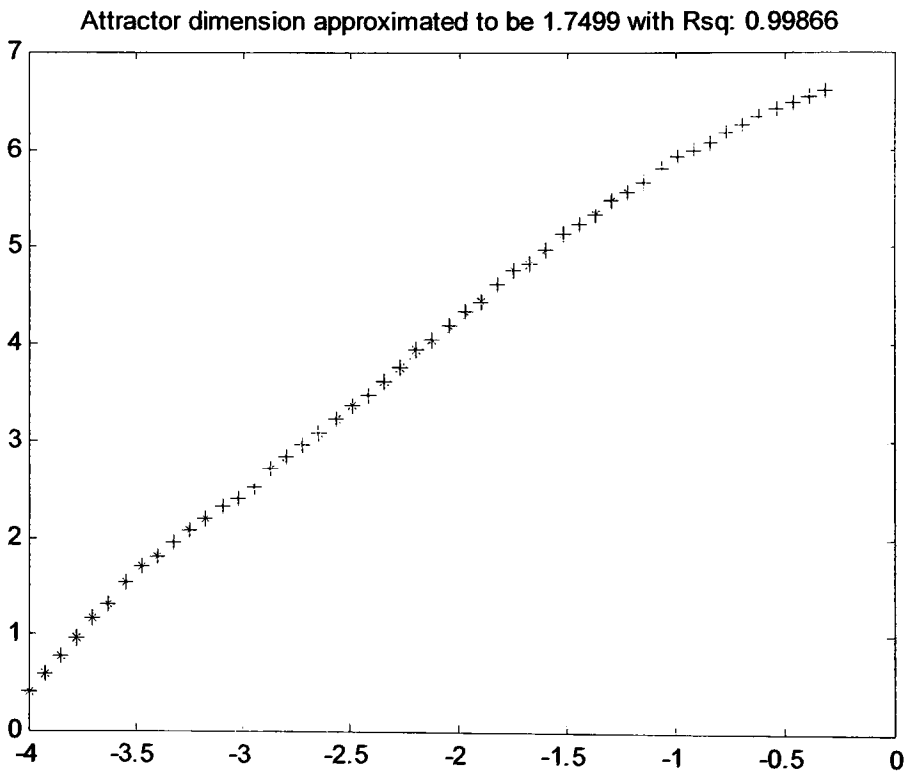
****

Program to calculate Information Dimension by Andrew Dick
11-Aug-2003
23:19:0.919

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Data set consisted of 1000 data points with a time step of 0.01 seconds
A transient period of 10 seconds was removed

The Information Dimension of the data set was determined to be 1.7499

0:1:41.951
Elapsed time 42 minutes, 41.032 seconds
Normal Termination of Program INFORMATION_DIMENSION.M
****



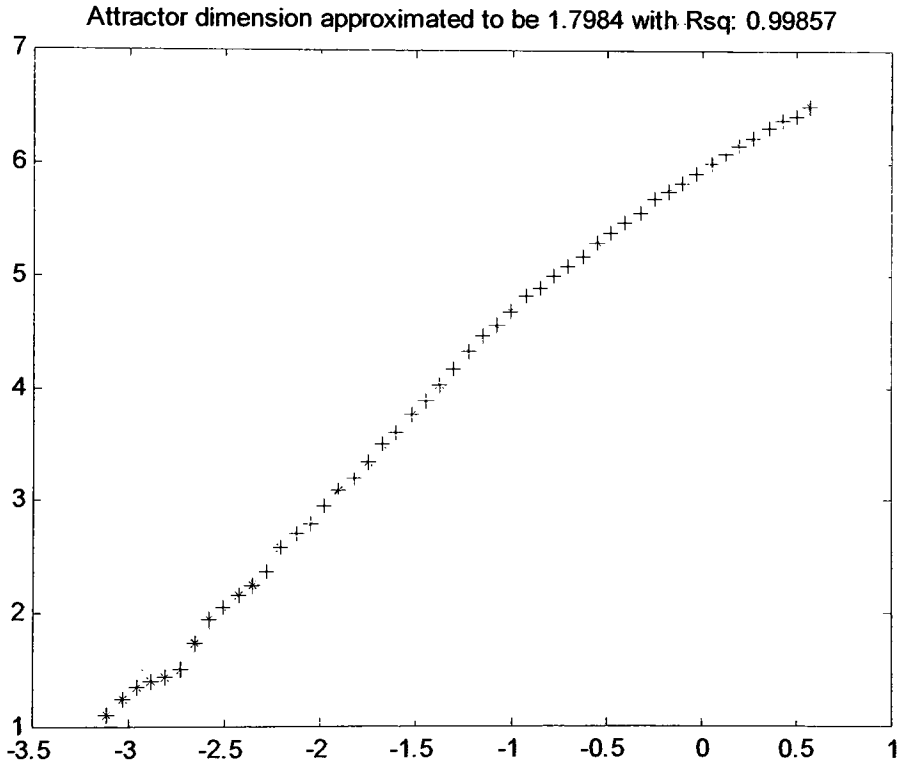
****

Program to calculate Information Dimension by Andrew Dick
12-Aug-2003
0:1:41.972

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Data set consisted of 1000 data points with a time step of 0.05 seconds
A transient period of 40 seconds was removed

The Information Dimension of the data set was determined to be 1.7984

1:15:16.509
Elapsed time 73 minutes, 34.537 seconds
Normal Termination of Program INFORMATION_DIMENSION.M
****



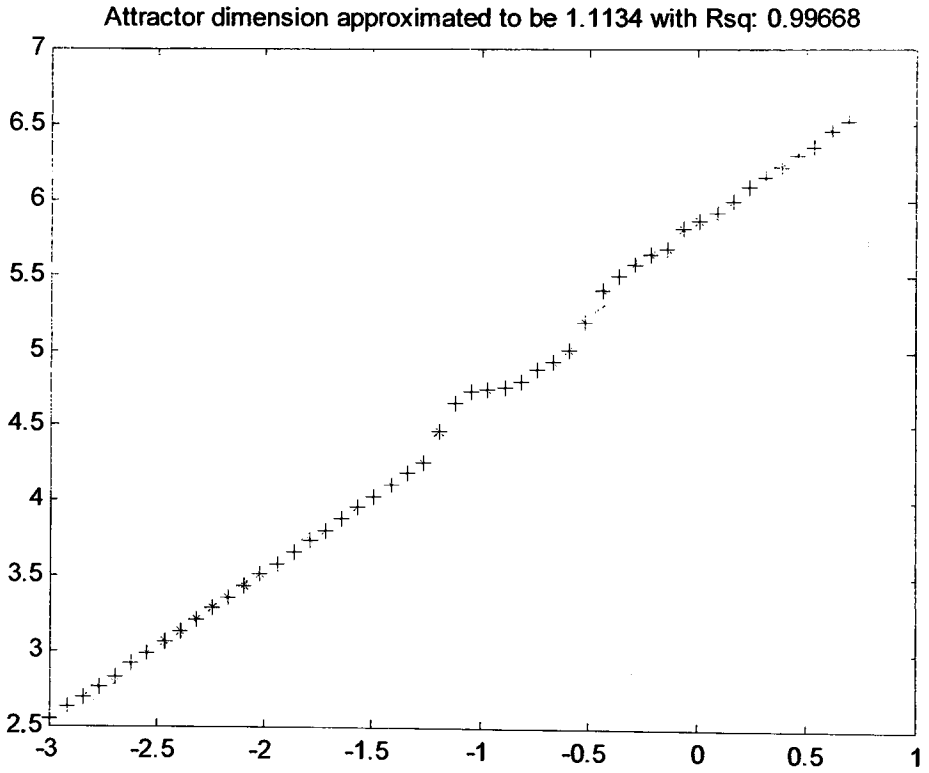
****

Program to calculate Information Dimension by Andrew Dick
13-Aug-2003
14:4:4.189

Data previously iterated using 5th order Lie Series approximation
Duffing system with Popular Parameter Set, $\epsilon = 0.25$, $F = 0.30$, $\omega = 1.0$
Data set consisted of 5000 data points with a time step of 0.05 seconds
A transient period of 20 seconds was removed

The Information Dimension of the data set was determined to be 1.1134

14:59:23.722
Elapsed time 55 minutes, 19.543 seconds
Normal Termination of Program INFORMATION_DIMENSION.M
****



APPENDIX C7

****

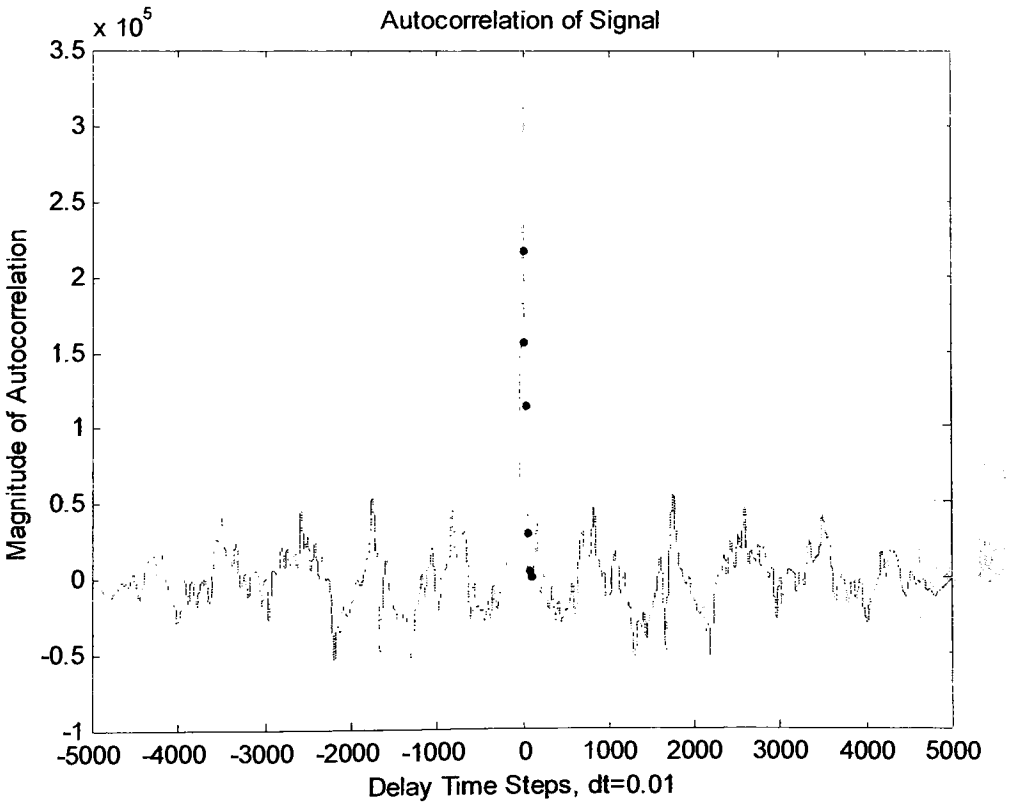
Delay Time value calculation program using autocorrelation by Andrew Dick
28-Jul-2003
9:26:29.671

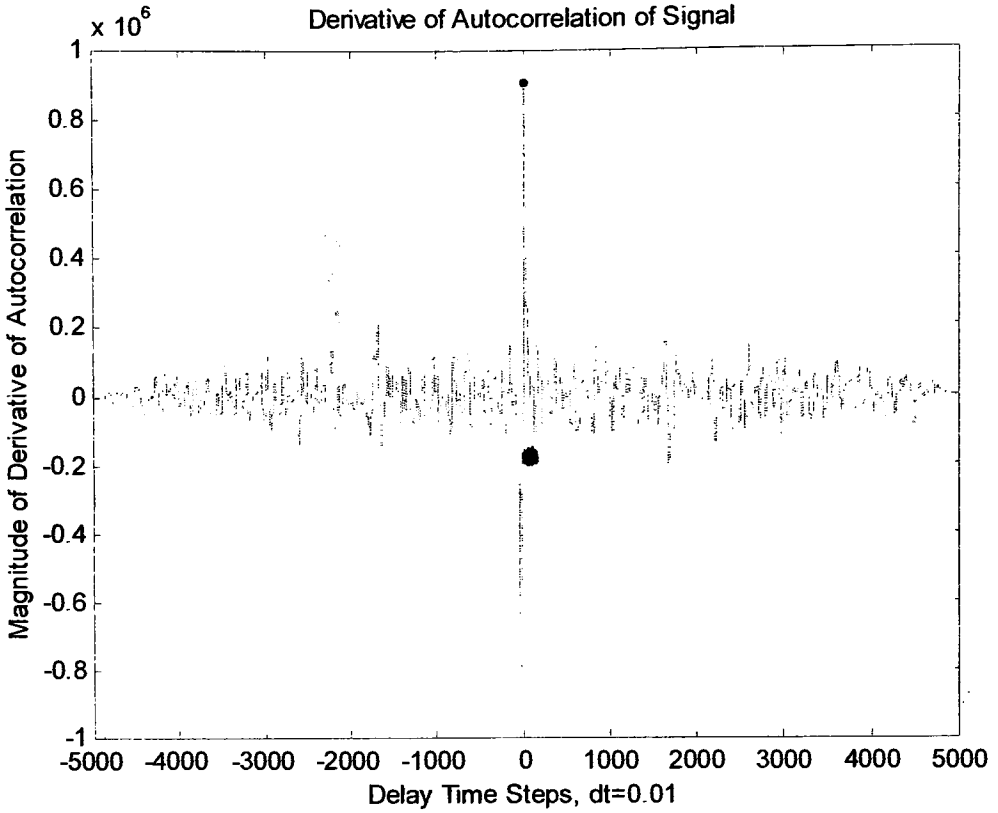
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Data set consisted of 5001 data points with a time step of 0.01 seconds
A transient period of 20 seconds was removed

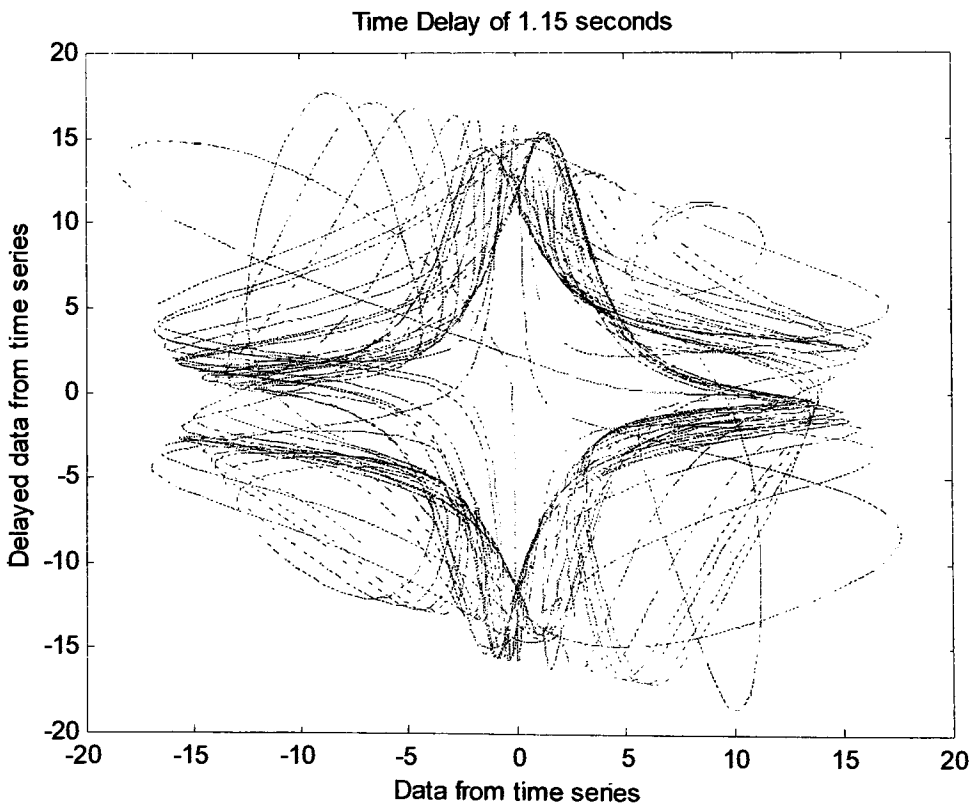
Delay Value at First Zero is 1.15 seconds
Delay Value at First Local Minimum is 0.8 seconds
Delay Value at Half of Maximum Value is 0.23 seconds
Delay Value at 1/e of Maximum Value is 0.29 seconds
Delay Value at One-Tenth of Maximum Value is 0.52 seconds
Delay Value at First Inflection Point is 0.16 seconds

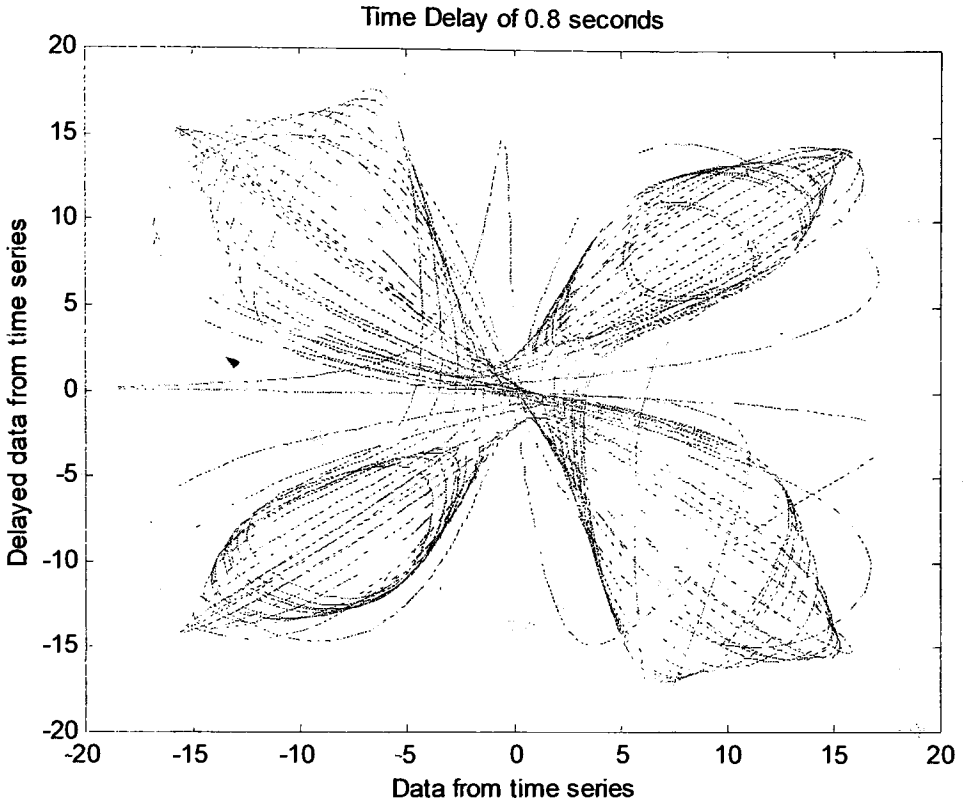
9:26:33.526
Elapsed time 0 minutes, 3.855 seconds
Normal Termination of Program AUTOCORRELATION_DELAY.M

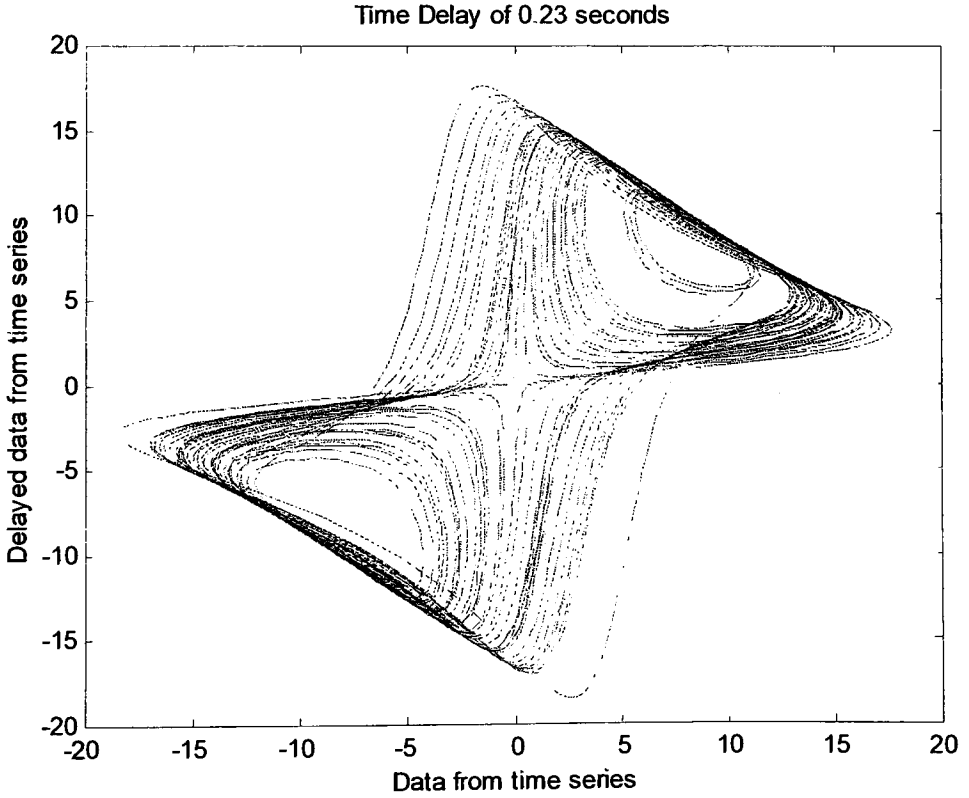
****

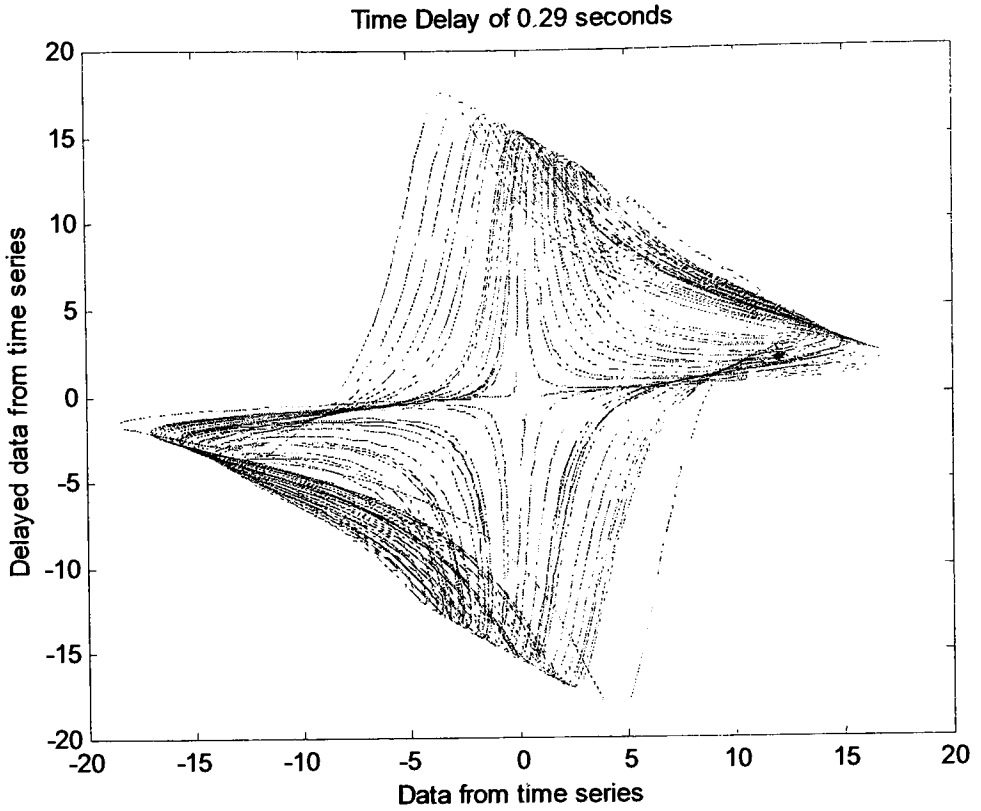


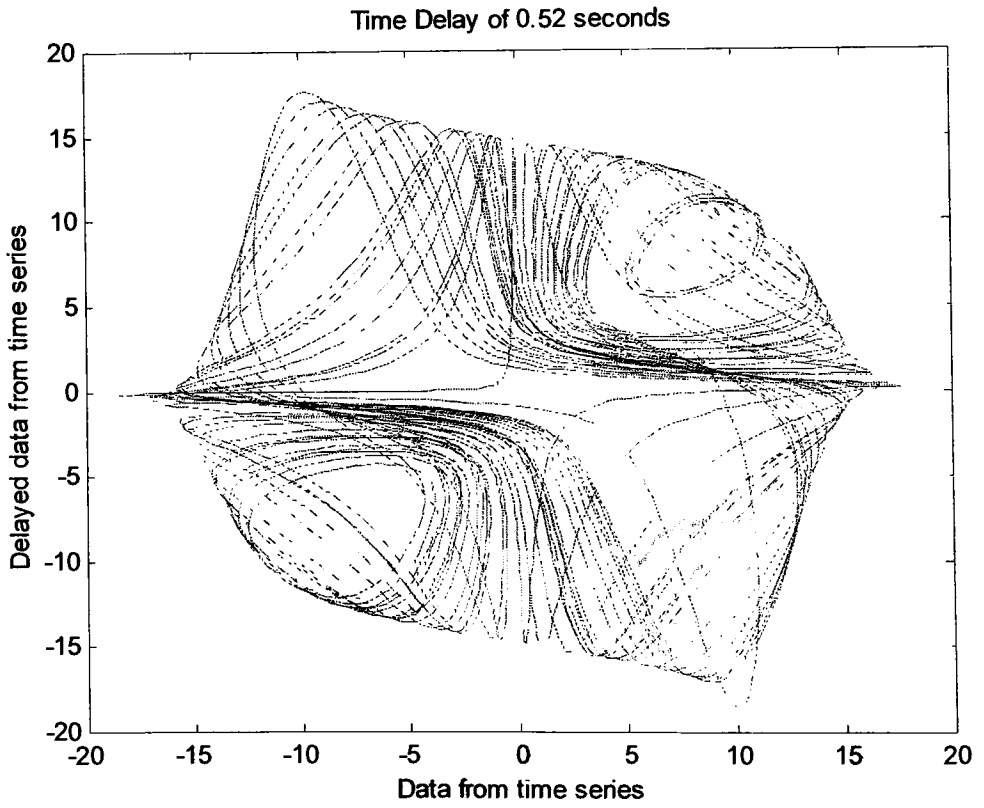

Derivative of Autocorrelation of Signal
x 10^6
Magnitude of Derivative of Autocorrelation
Delay Time Steps, dt=0.01

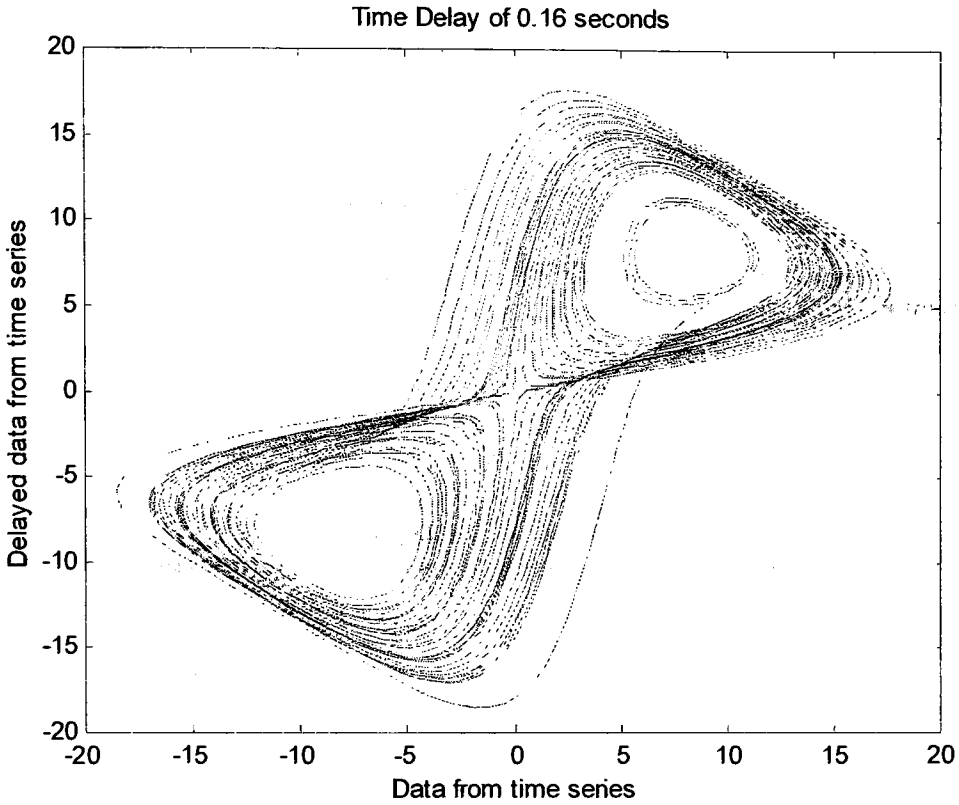

Time Delay of 1.15 seconds
Delayed data from time series
Data from time series


Time Delay of 0.8 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-20
-20
-15
-10
-5
0
5
10
15
20


Time Delay of 0.23 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-20
-20
-15
-10
-5
0
5
10
15
20


Time Delay of 0.29 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-20
-20
-15
-10
-5
0
5
10
15
20


Time Delay of 0.52 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-20
-20
-15
-10
-5
0
5
10
15
20


Time Delay of 0.16 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-20
-20
-15
-10
-5
0
5
10
15
20

****

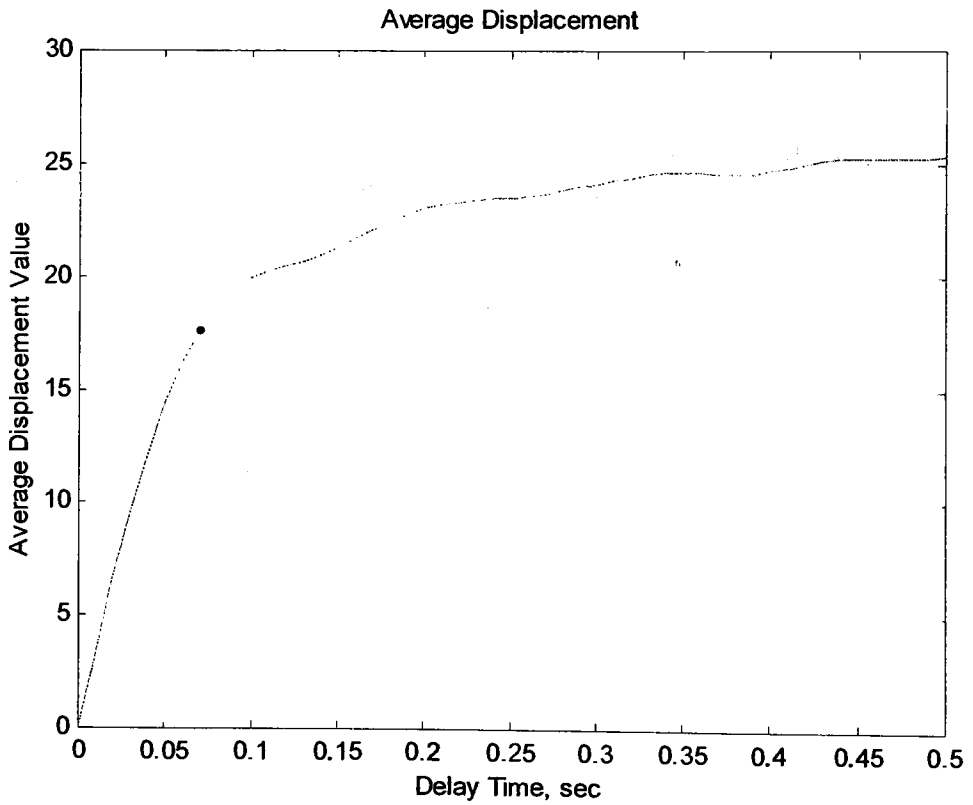
Delay Time value calculation program using average displacement by Andrew Dick
28-Jul-2003
9:26:35.459

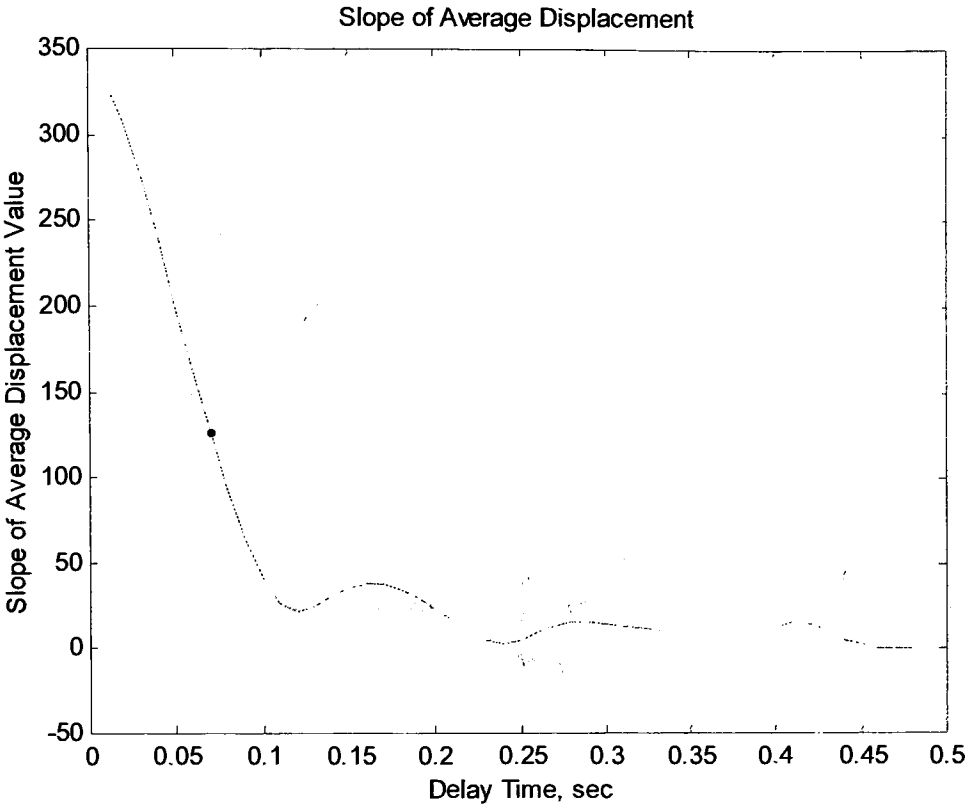
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Data set consisted of 5000 data points with a time step of 0.01 seconds
A transient period of 20 seconds was removed

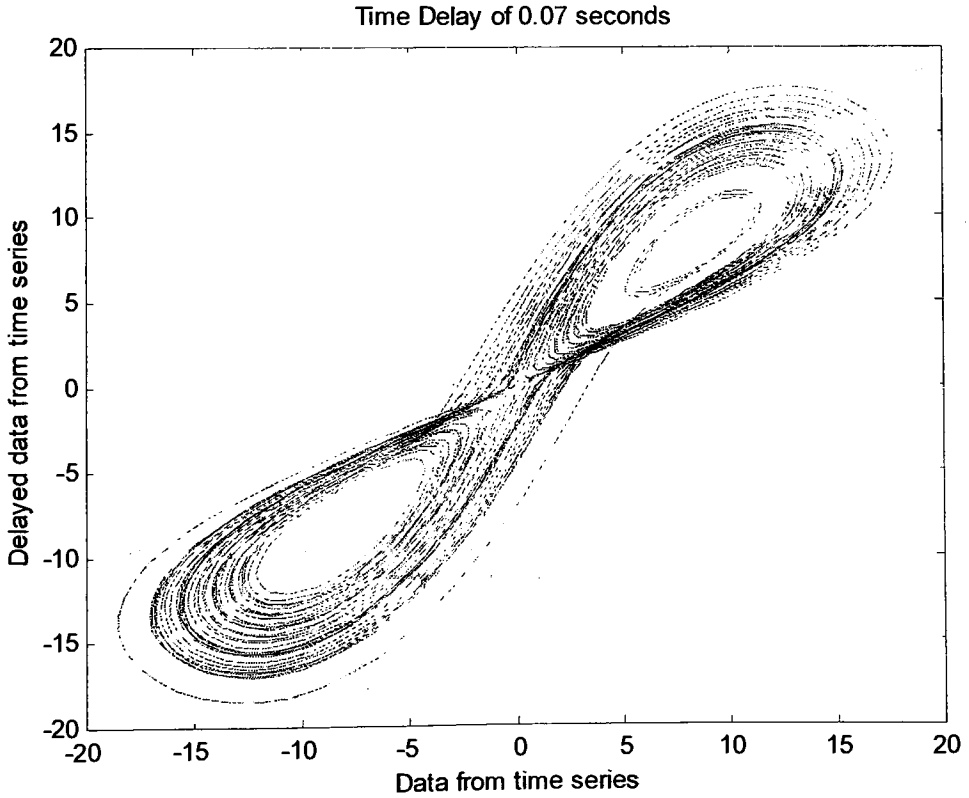
Delay Value of 0.07 seconds at 40% of slope at S(1)

9:26:45.744
Elapsed time 0 minutes, 10.295 seconds
Normal Termination of Program AVERAGE_DISPLACEMENT.M

****




Slope of Average Displacement
Slope of Average Displacement Value
Delay Time, sec
350
300
250
200
150
100
50
0
-50
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Time Delay of 0.07 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-20
-20
-15
-10
-5
0
5
10
15
20

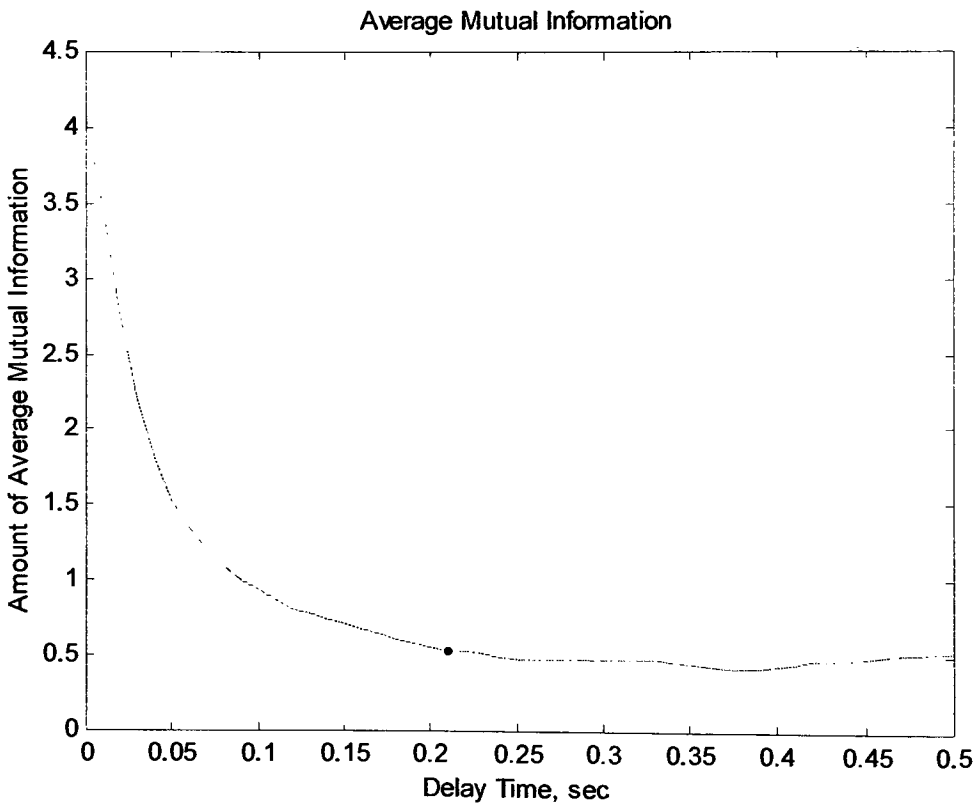
****

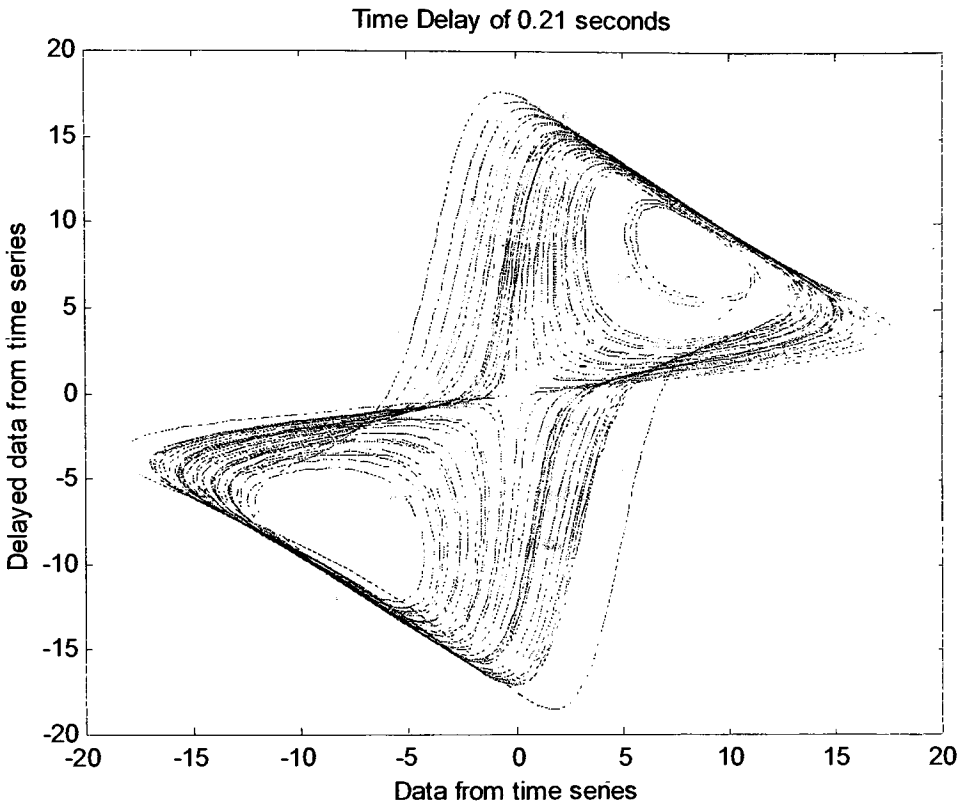
Delay Time value calculation program using Average Mutual Information by Andrew Dick
28-Jul-2003
9:26:45.744

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Data set consisted of 5000 data points with a time step of 0.01 seconds
A transient period of 20 seconds was removed

Delay Value for First Local Minimum is 0.21 seconds

9:30:35.915
Elapsed time 3 minutes, 50.171 seconds
Normal Termination of Program MUTUAL_INFORMATION.M
****



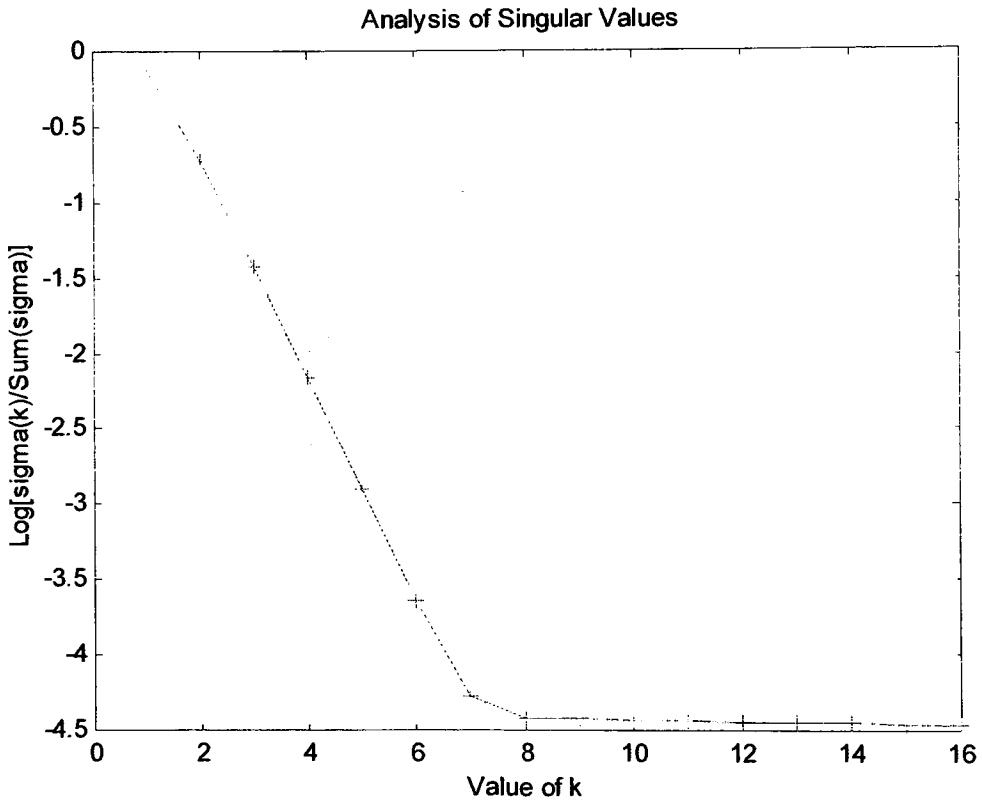

Time Delay of 0.21 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-20
-20
-15
-10
-5
0
5
10
15
20

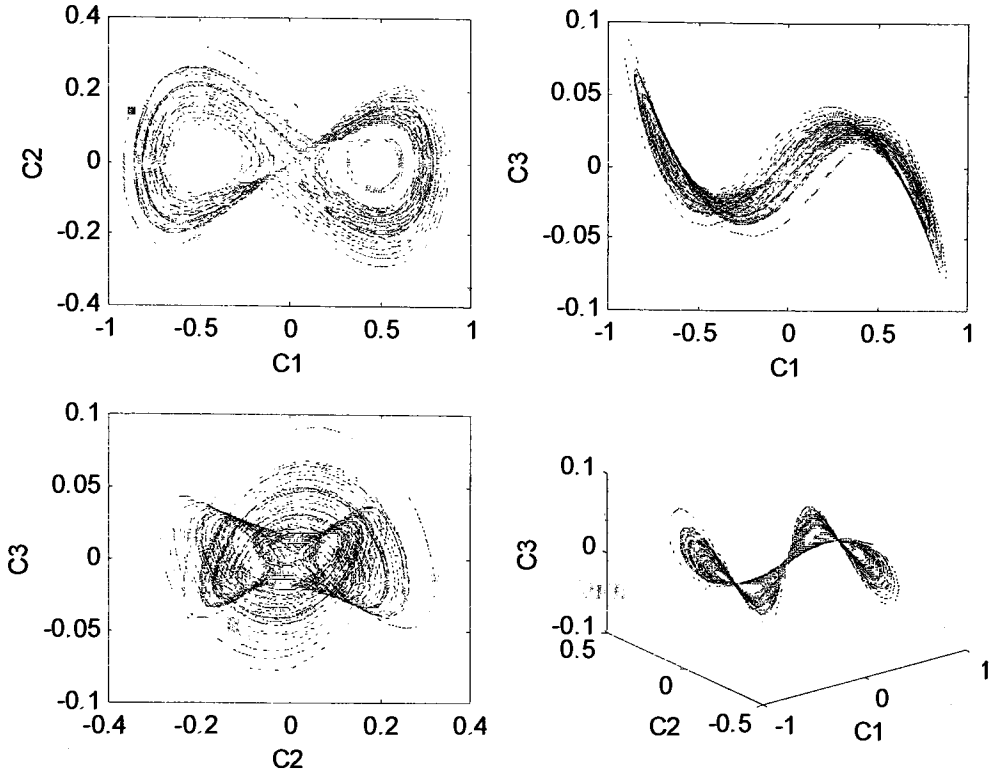
****

Singular System Approach for attractor reconstruction by Andrew Dick
28-Jul-2003
9:33:49.473

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
5000 data points used with time step of 0.01
20 seconds of transient data removed

9:34:12.637
Elapsed time 0 minutes, 23.164 seconds
Normal Termination of Program SINGULAR_SYSTEM_APPROACH.M
****




0.4
0.2
0
-0.2
-0.4
C2
-1
-0.5
0
0.5
1
C1
0.1
0.05
0
-0.05
-0.1
C3
-1
-0.5
0
0.5
1
C1
0.1
0.05
0
-0.05
-0.1
C3
-0.4
-0.2
0
0.2
0.4
C2
0.1
0
-0.1
C3
0.5
0
C2
-0.5
-1
0
C1
1

APPENDIX C8

****

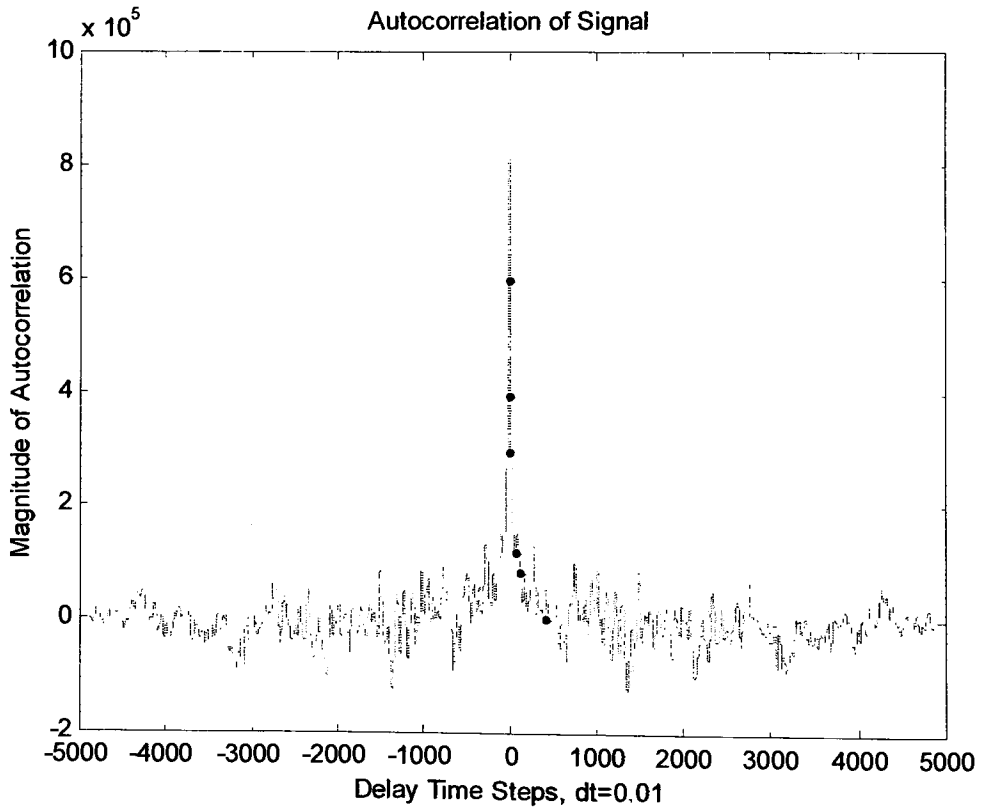
Delay Time value calculation program using autocorrelation by Andrew Dick
28-Jul-2003
12:38:43.62

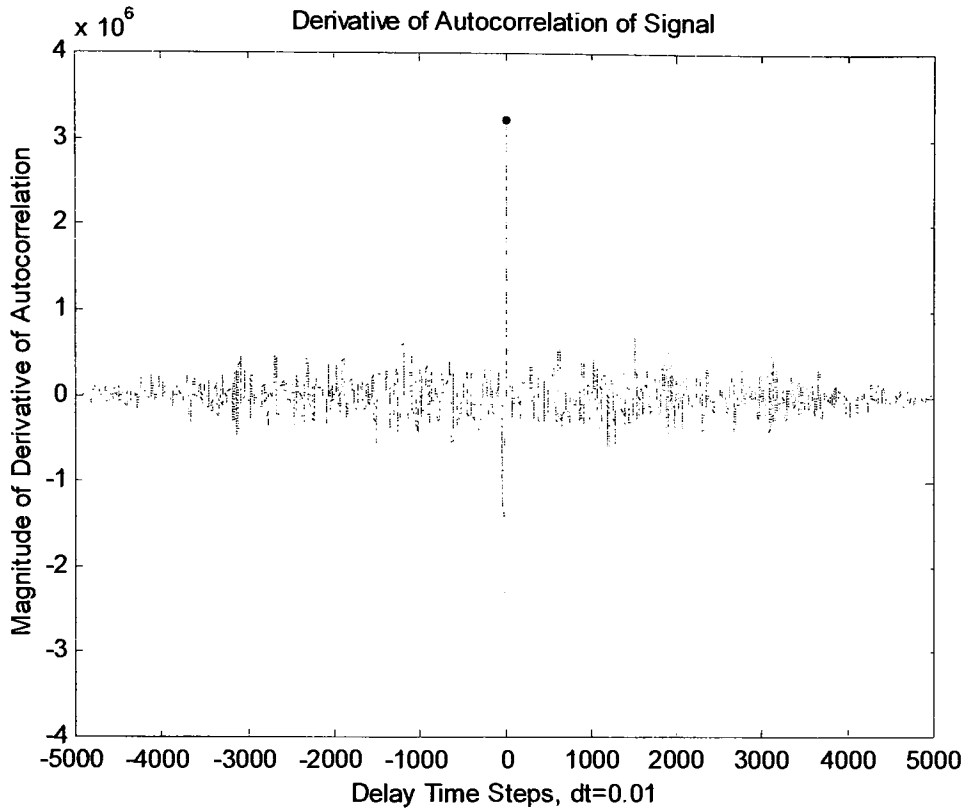
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Data set consisted of 5001 data points with a time step of 0.01 seconds
A transient period of 10 seconds was removed

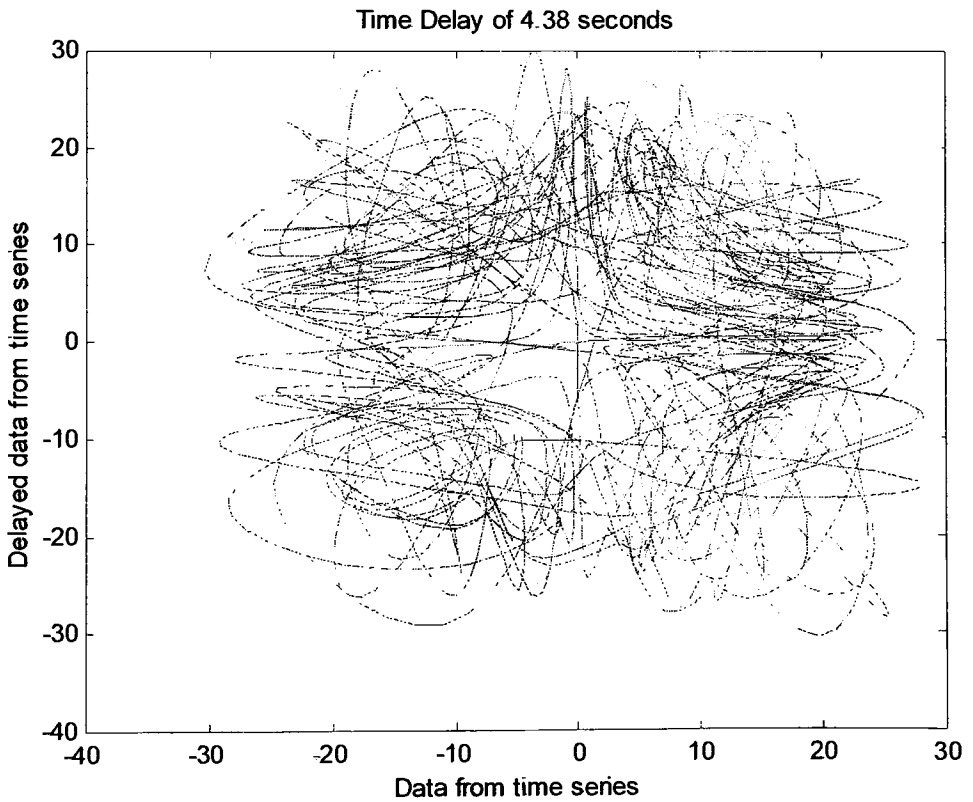
Delay Value at First Zero is 4.38 seconds
Delay Value at First Local Minimum is 0.75 seconds
Delay Value at Half of Maximum Value is 0.17 seconds
Delay Value at 1/e of Maximum Value is 0.22 seconds
Delay Value at One-Tenth of Maximum Value is 1.32 seconds
Delay Value at First Inflection Point is 0.1 seconds

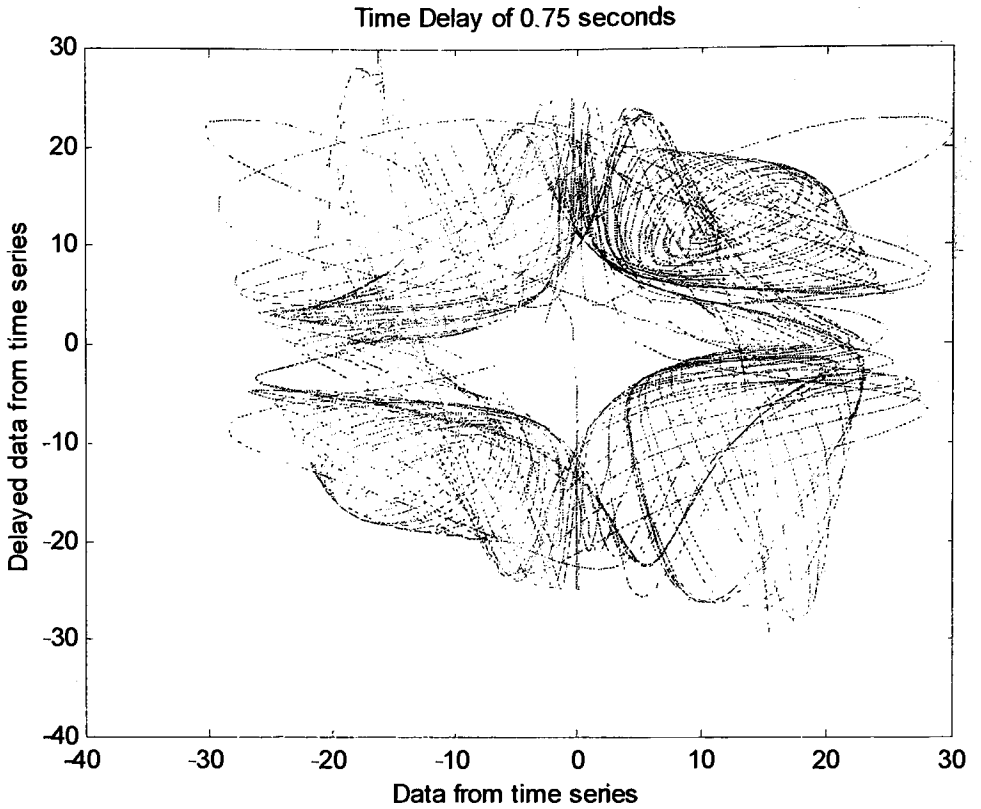
12:38:47.816
Elapsed time 0 minutes, 4.196 seconds
Normal Termination of Program AUTOCORRELATION_DELAY.M

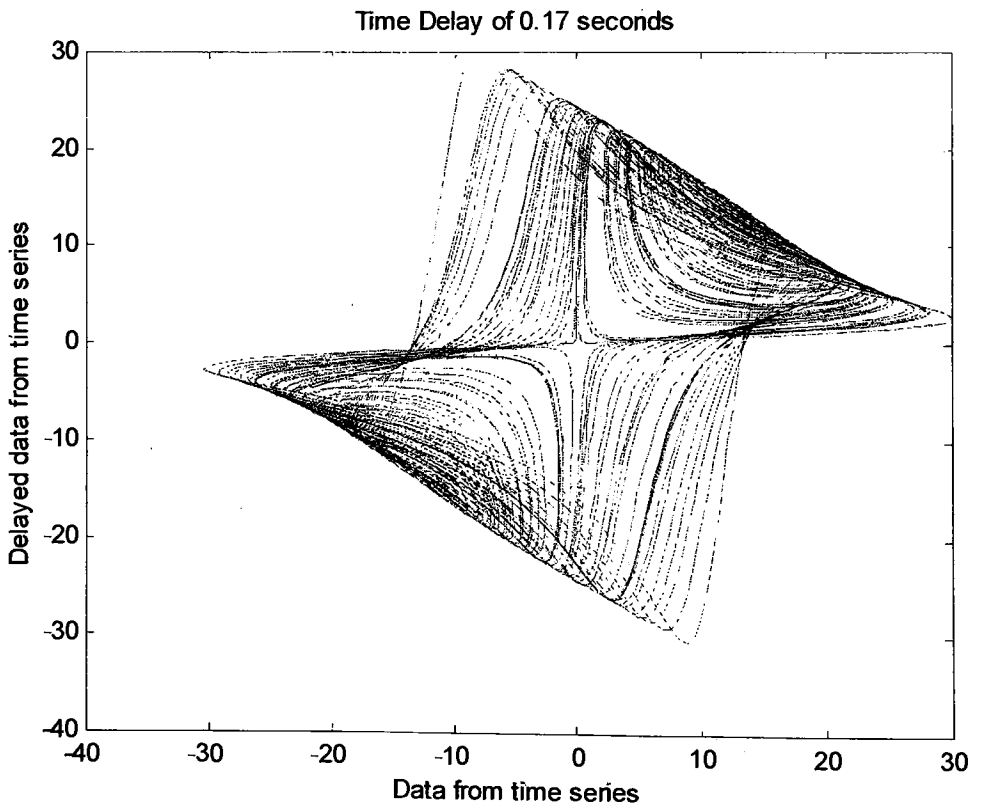
****

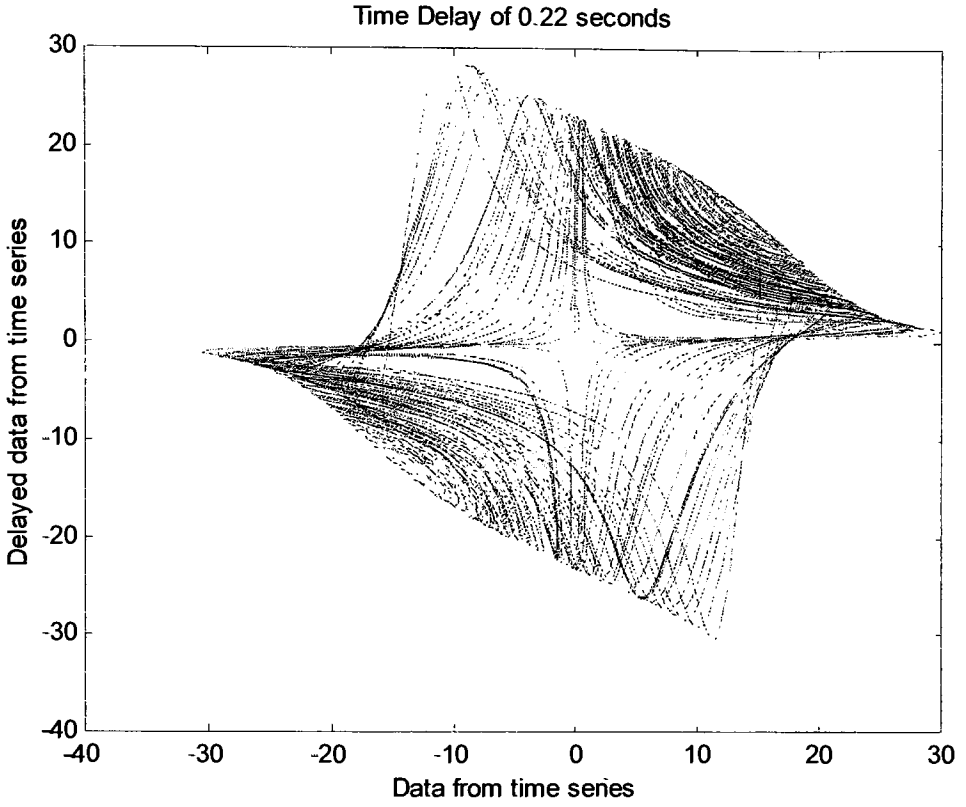


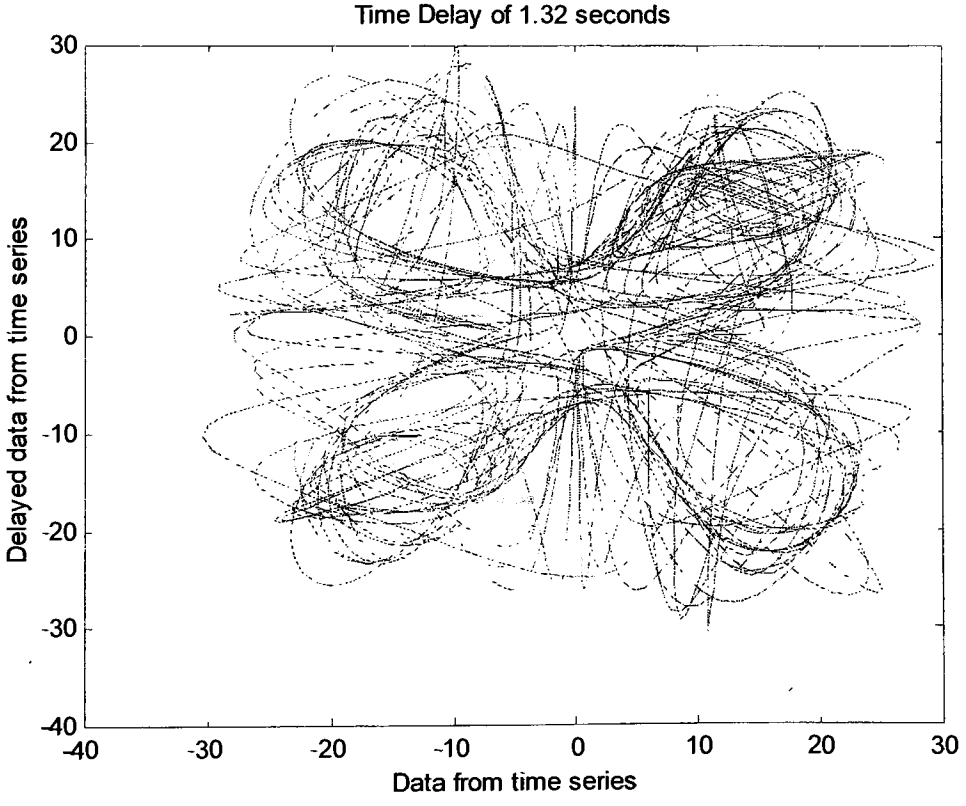

x 10^6
Derivative of Autocorrelation of Signal
Magnitude of Derivative of Autocorrelation
Delay Time Steps, dt=0.01
4
3
2
1
0
-1
-2
-3
-4
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

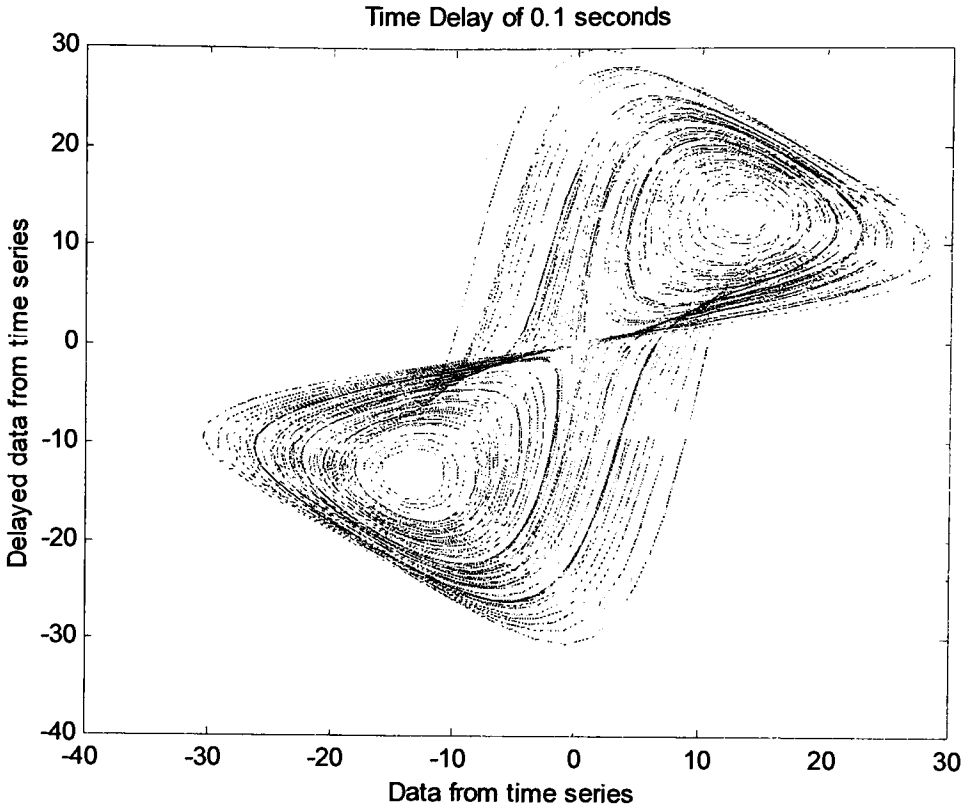

Time Delay of 4.38 seconds
Delayed data from time series
Data from time series
30
20
10
0
-10
-20
-30
-40


Time Delay of 0.75 seconds
Delayed data from time series
Data from time series
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30


Time Delay of 0.17 seconds
Delayed data from time series
Data from time series
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30


Time Delay of 0.22 seconds
Delayed data from time series
Data from time series
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30


Time Delay of 1.32 seconds
Delayed data from time series
Data from time series
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30


Time Delay of 0.1 seconds
Delayed data from time series
Data from time series
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30

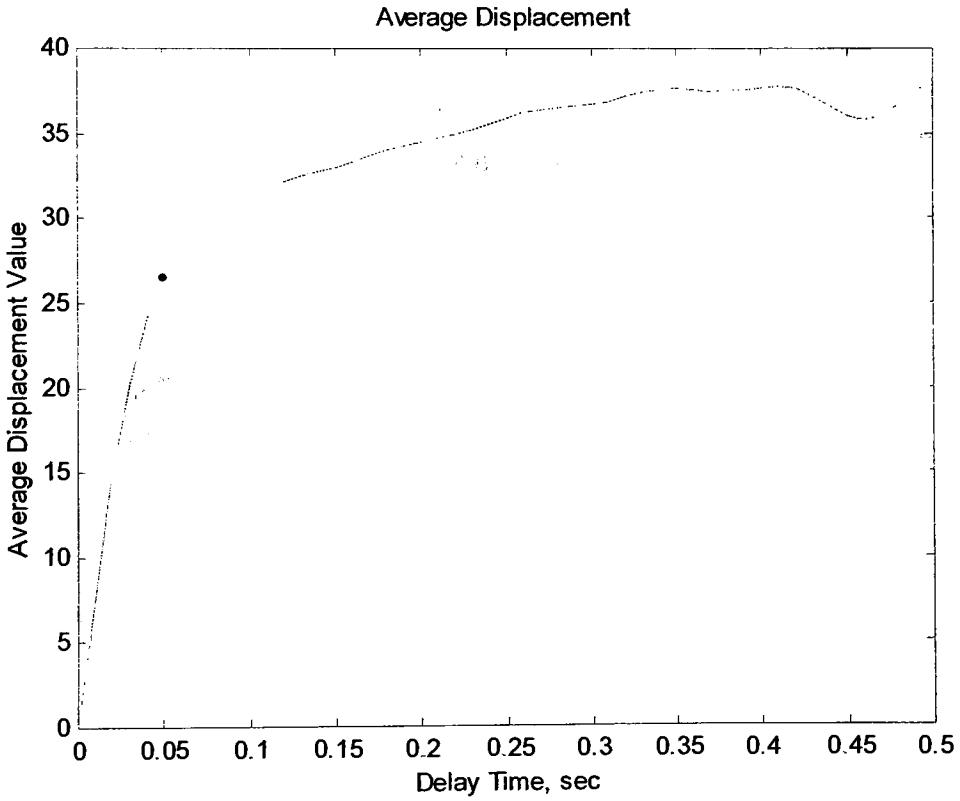
****

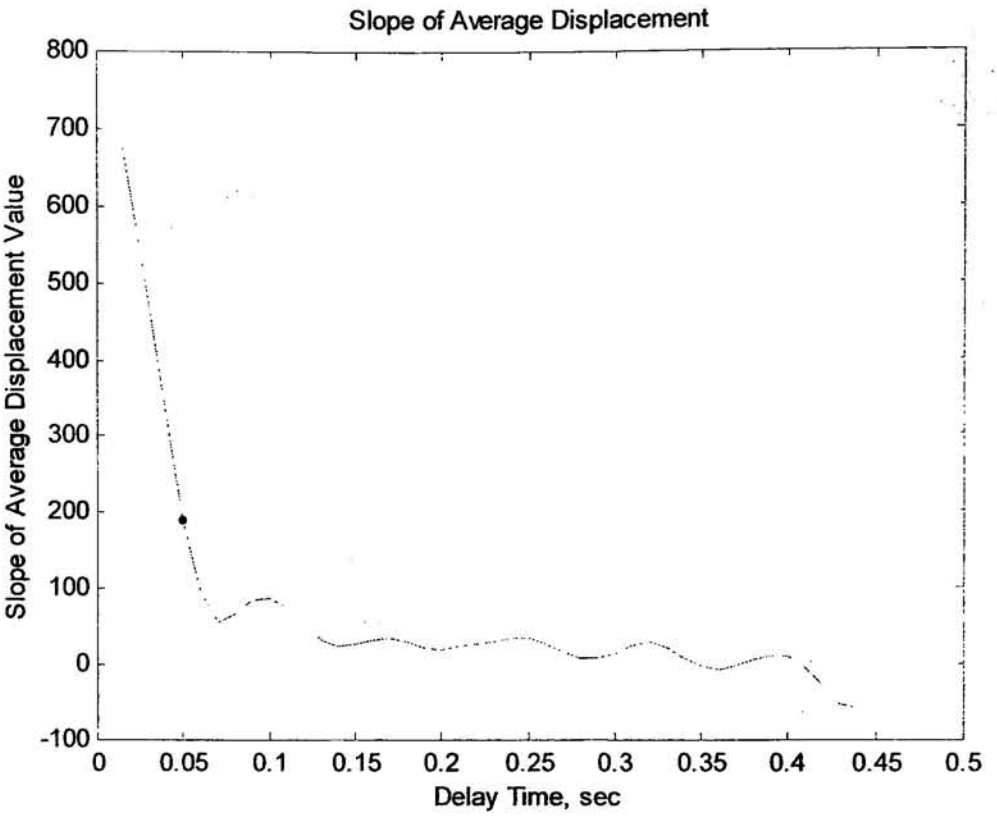
Delay Time value calculation program using average displacement by Andrew Dick
28-Jul-2003
12:38:49.939

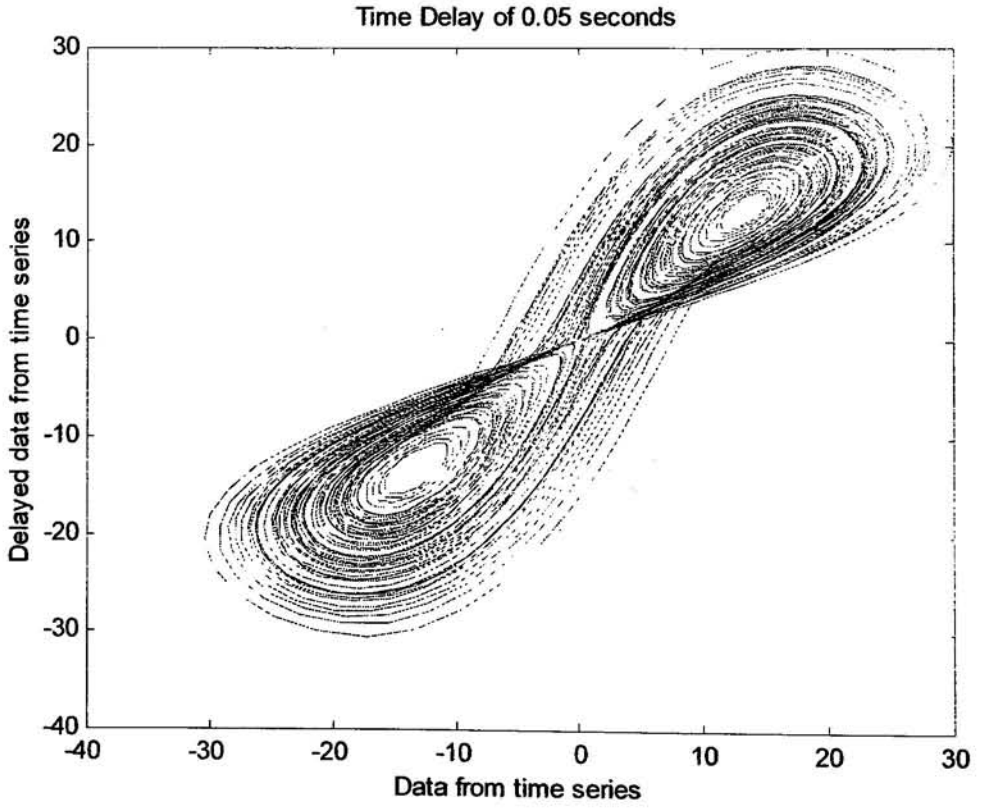
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Data set consisted of 5000 data points with a time step of 0.01 seconds
A transient period of 10 seconds was removed

Delay Value of 0.05 seconds at 40% of slope at S(1)

12:39:0.434
Elapsed time 0 minutes, 10.495 seconds
Normal Termination of Program AVERAGE_DISPLACEMENT.M
****




Slope of Average Displacement
Slope of Average Displacement Value
800
700
600
500
400
300
200
100
0
-100
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
Delay Time, sec


Time Delay of 0.05 seconds
Delayed data from time series
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30
Data from time series

****

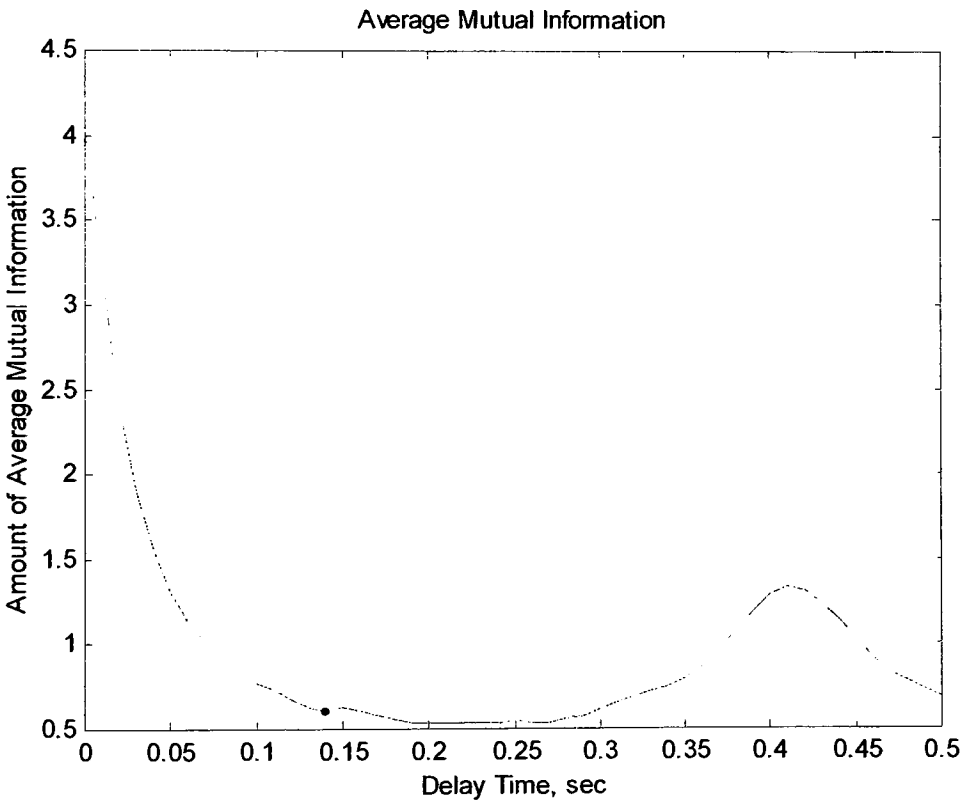
Delay Time value calculation program using Average Mutual Information by Andrew Dick
28-Jul-2003
12:39:0.444

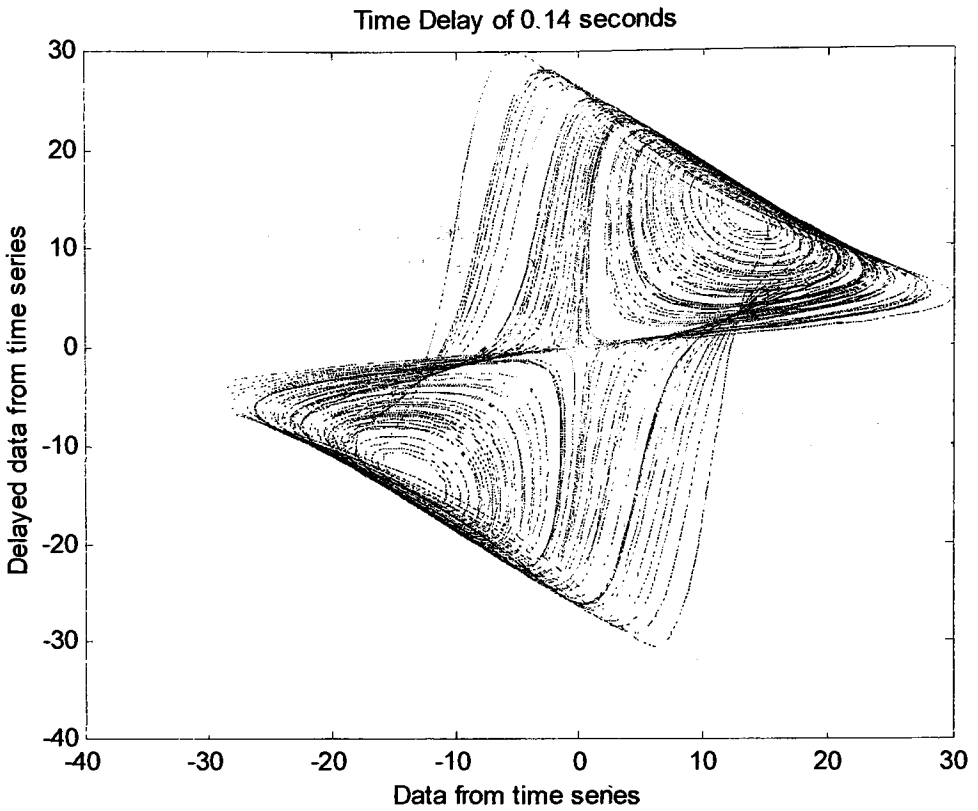
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Data set consisted of 5000 data points with a time step of 0.01 seconds
A transient period of 10 seconds was removed

Delay Value for First Local Minimum is 0.14 seconds

12:43:59.054
Elapsed time 4 minutes, 58.61 seconds
Normal Termination of Program MUTUAL_INFORMATION.M

****



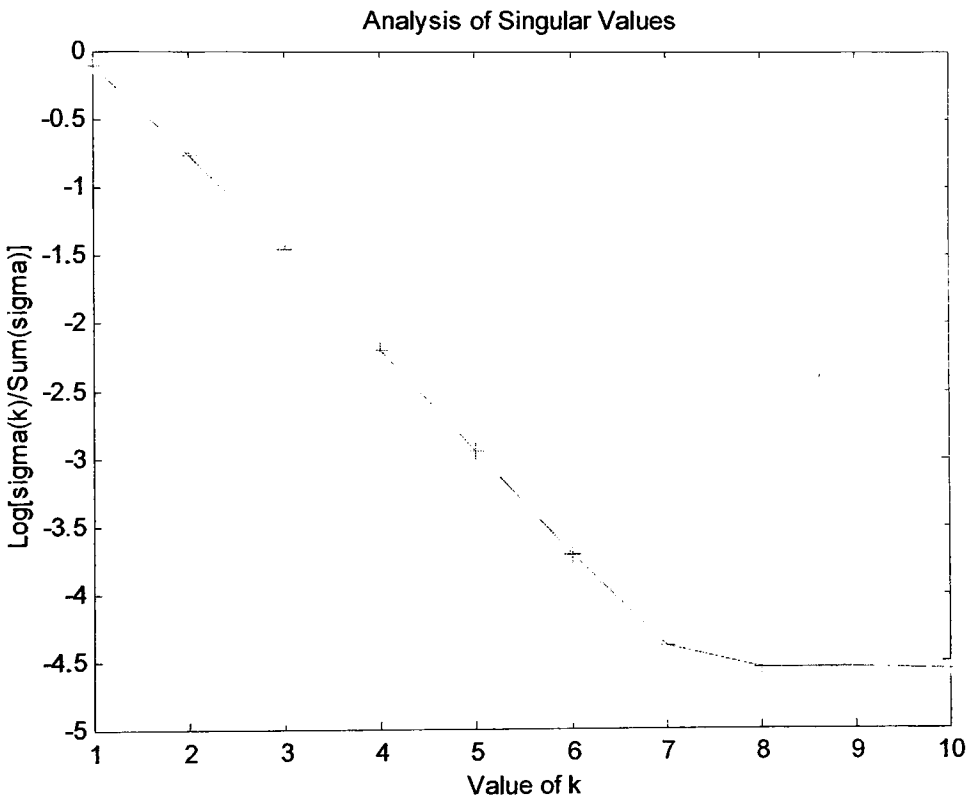

Time Delay of 0.14 seconds
Delayed data from time series
Data from time series
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30

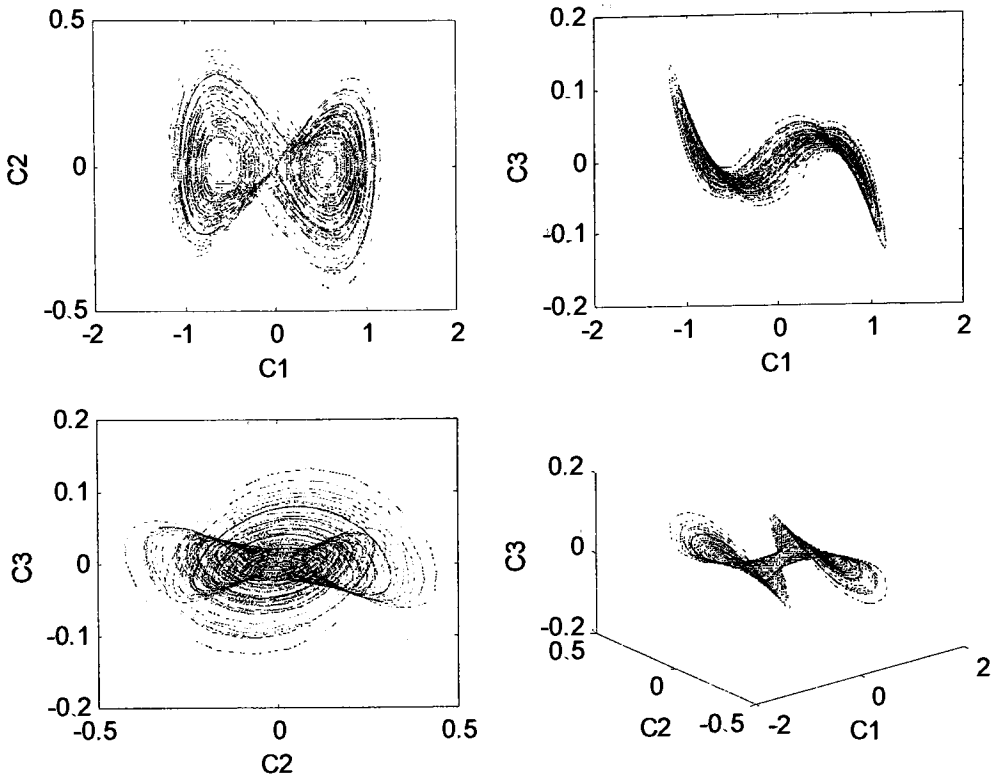
****

Singular System Approach for attractor reconstruction by Andrew Dick
28-Jul-2003
12:46:53.555

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
5000 data points used with time step of 0.01
10 seconds of transient data removed

12:47:4.43
Elapsed time 0 minutes, 10.885 seconds
Normal Termination of Program SINGULAR_SYSTEM_APPROACH.M
****




0.5
0
-0.5
C2
-2
-1
0
1
2
C1
0.2
0.1
0
-0.1
-0.2
C3
-2
-1
0
1
2
C1
0.2
0.1
0
-0.1
-0.2
C3
-0.5
0
0.5
C2
0.2
0
-0.2
C3
0.5
0
C2
-0.5
-2
0
C1
2

APPENDIX C9

****

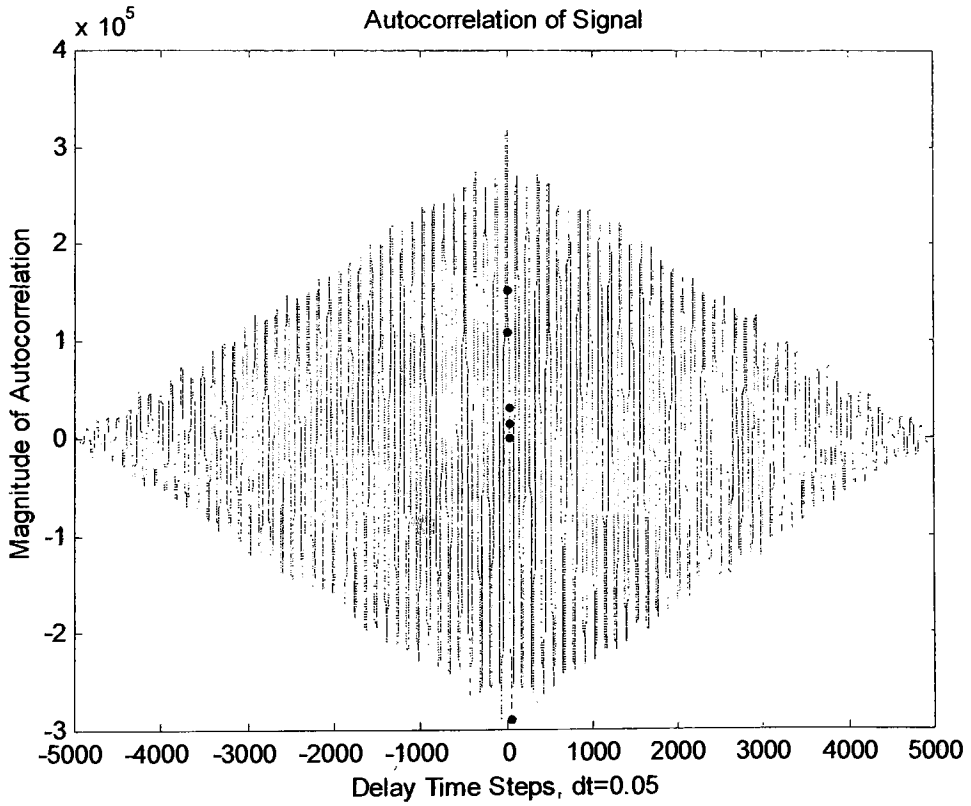
Delay Time value calculation program using autocorrelation by Andrew Dick
28-Jul-2003
14:9:49.57

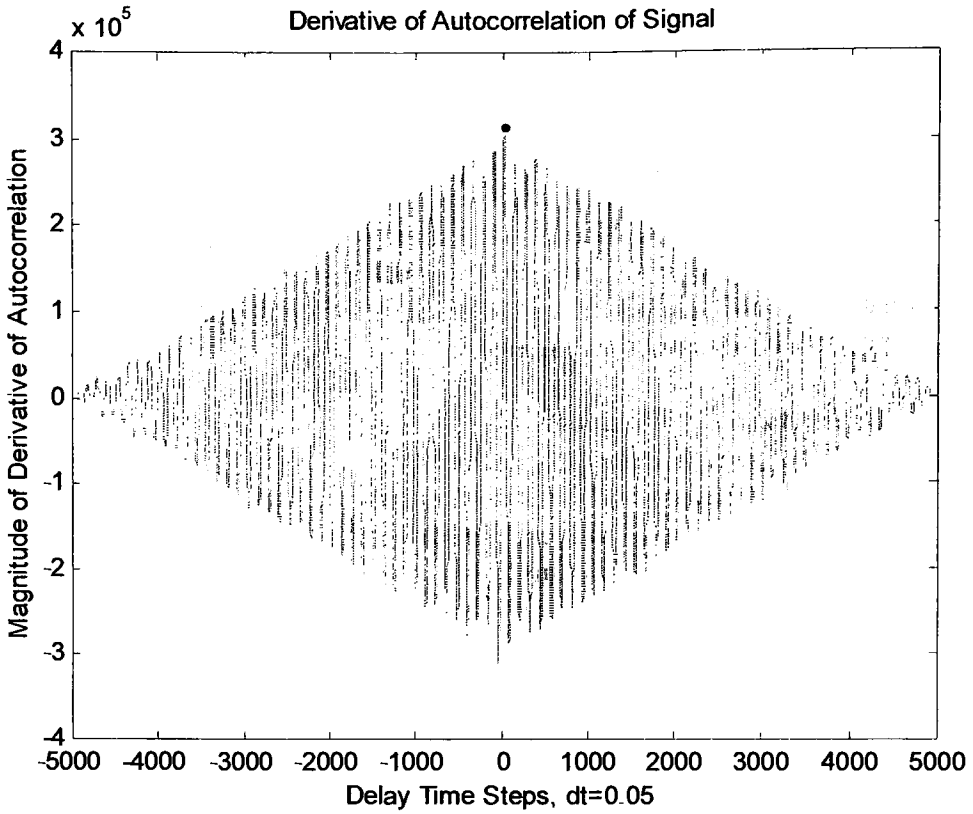
Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Data set consisted of 5001 data points with a time step of 0.05 seconds
A transient period of 40 seconds was removed

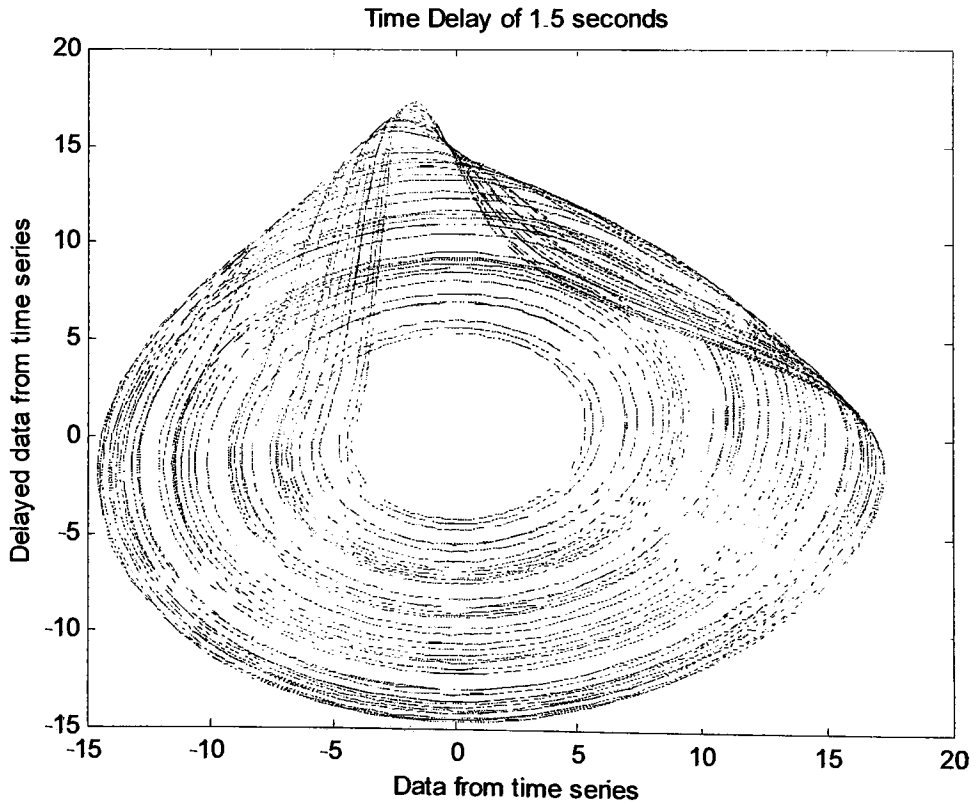
Delay Value at First Zero is 1.5 seconds
Delay Value at First Local Minimum is 3 seconds
Delay Value at Half of Maximum Value is 1 seconds
Delay Value at 1/e of Maximum Value is 1.15 seconds
Delay Value at One-Tenth of Maximum Value is 1.4 seconds
Delay Value at First Inflection Point is 1.45 seconds

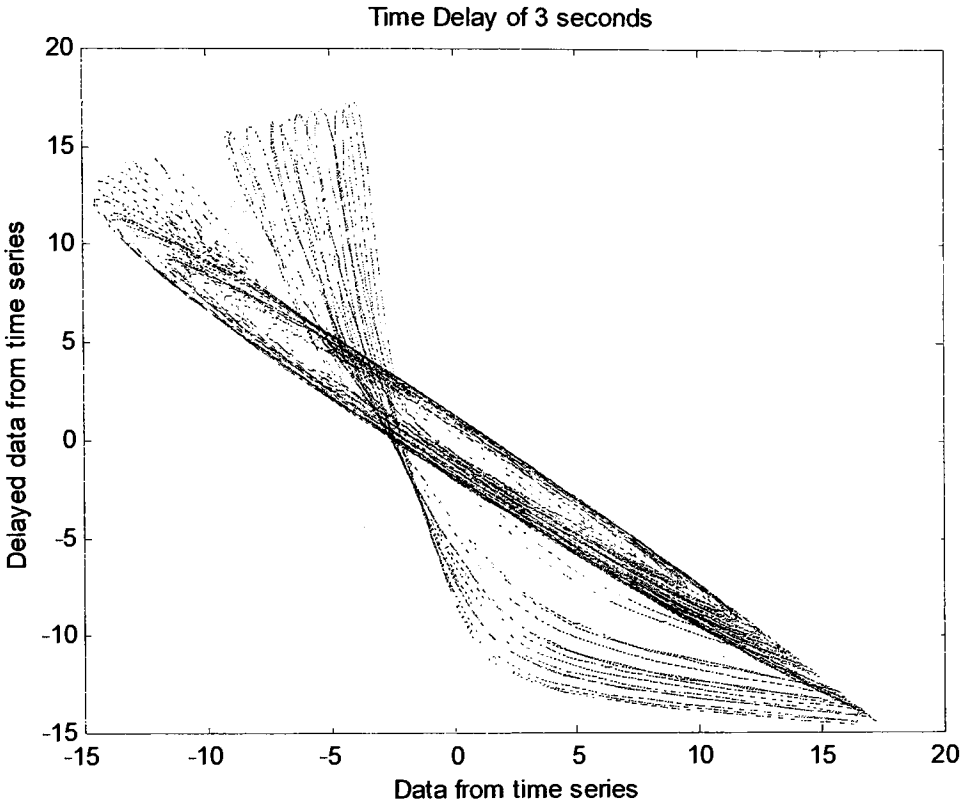
14:9:54.807
Elapsed time 0 minutes, 5.237 seconds
Normal Termination of Program AUTOCORRELATION_DELAY.M

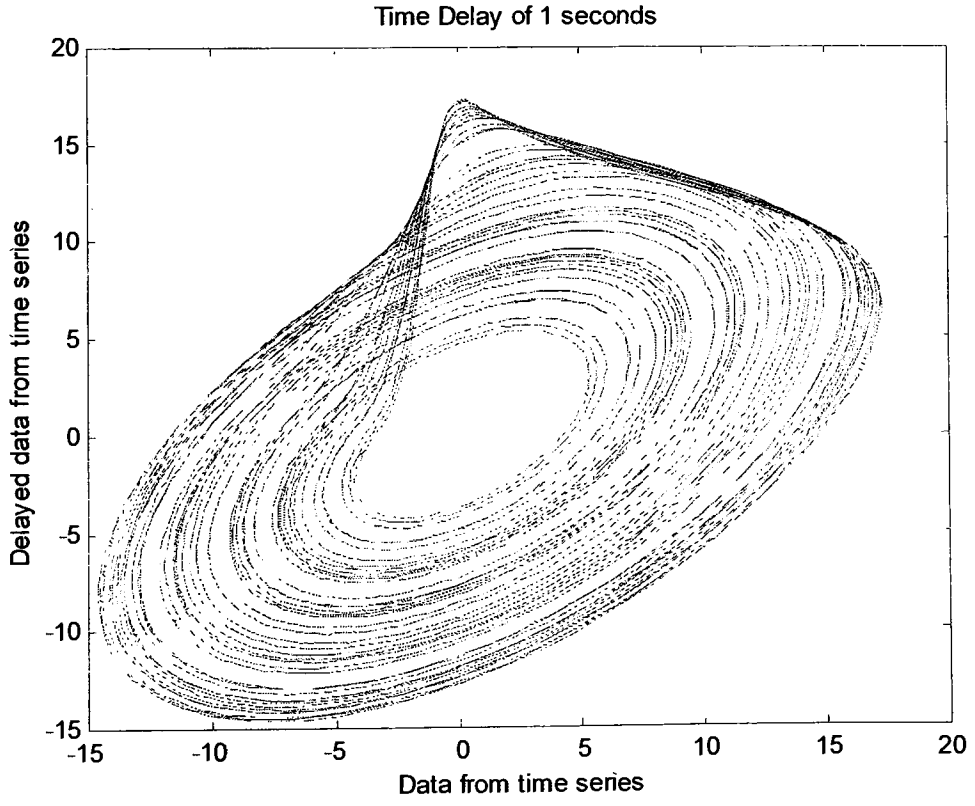
****

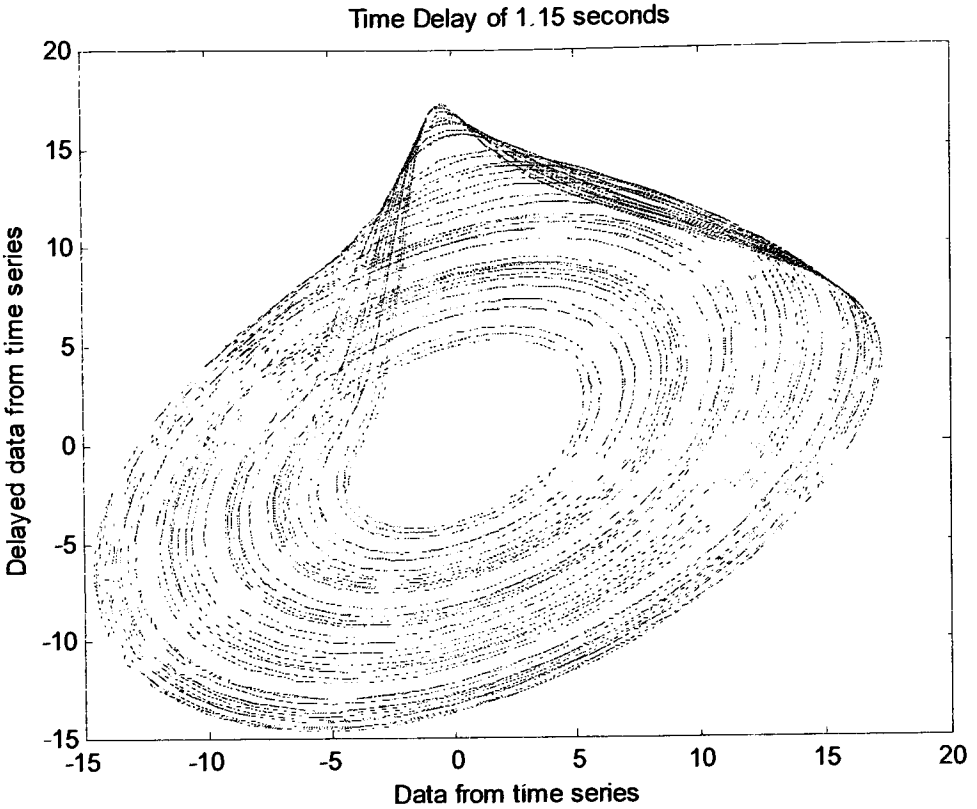


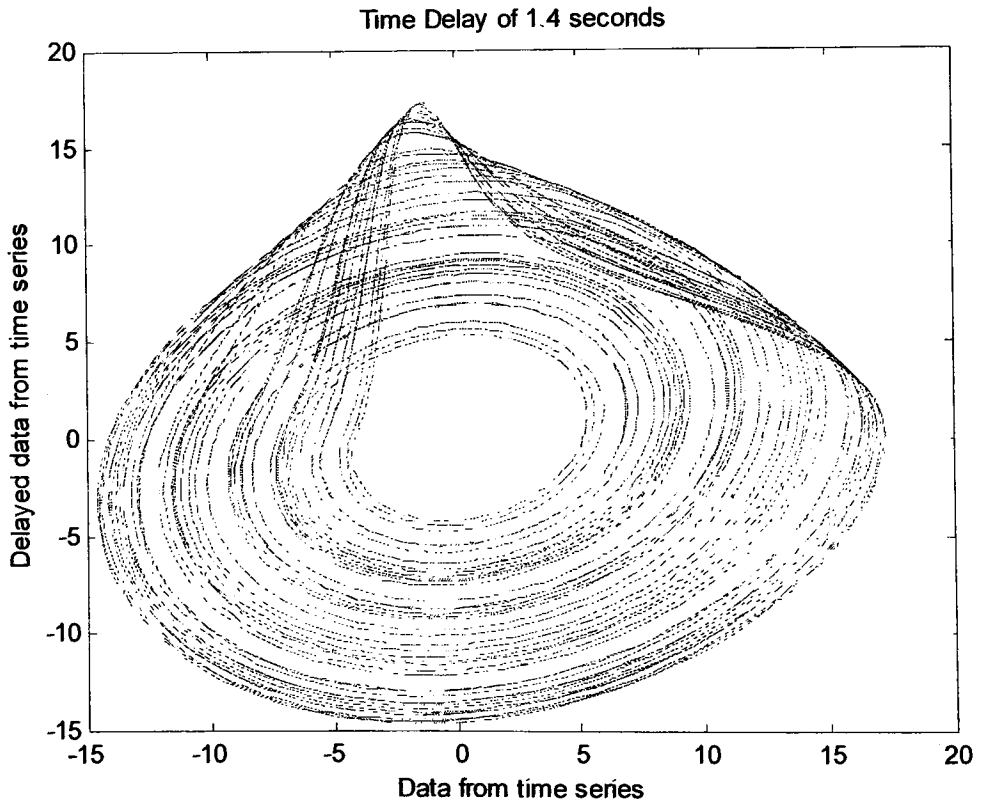

Derivative of Autocorrelation of Signal
x 10^5
Magnitude of Derivative of Autocorrelation
Delay Time Steps, dt=0.05

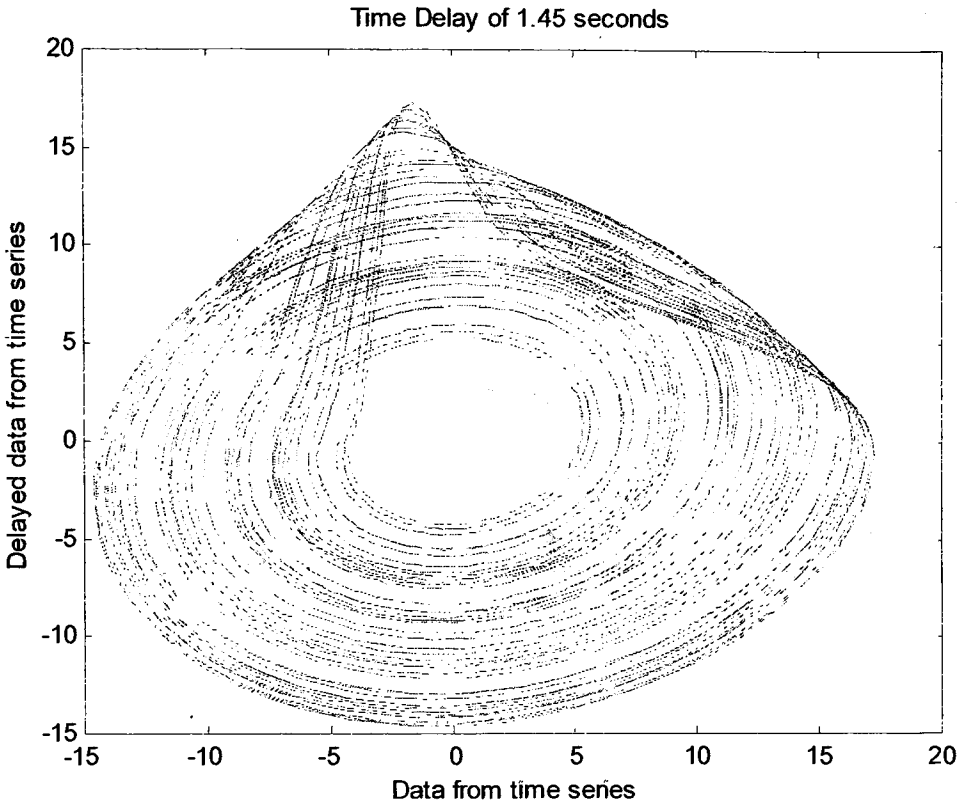

Time Delay of 1.5 seconds
Delayed data from time series
Data from time series


Time Delay of 3 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20


Time Delay of 1 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20


Time Delay of 1.15 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20


Time Delay of 1.4 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20


Time Delay of 1.45 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20

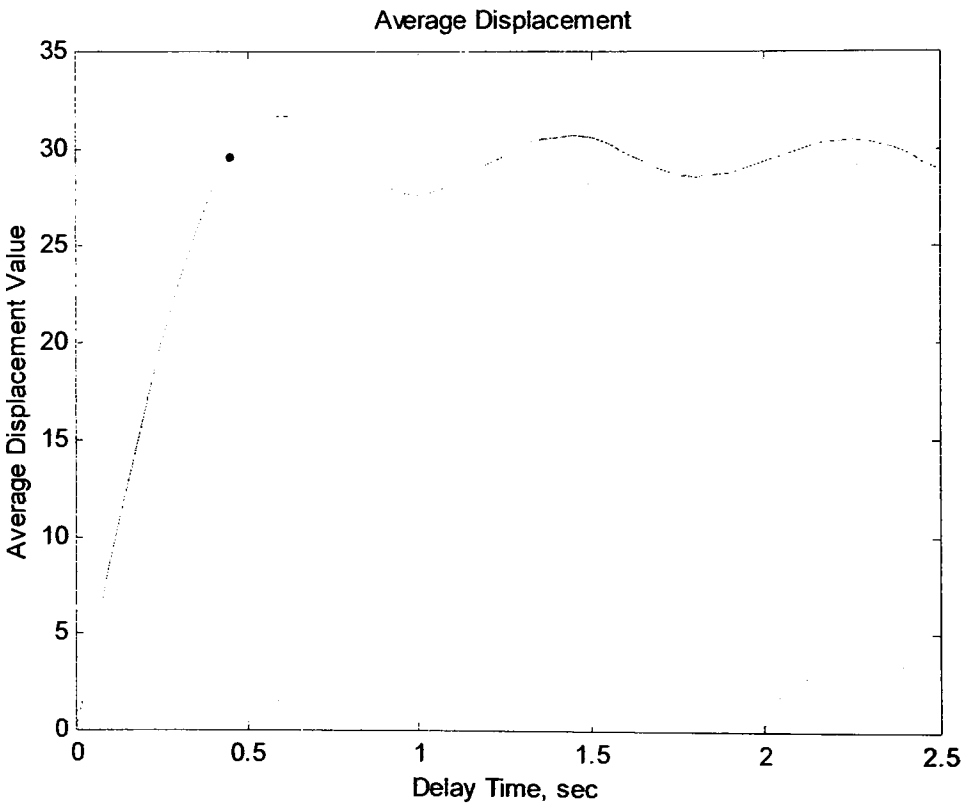
****

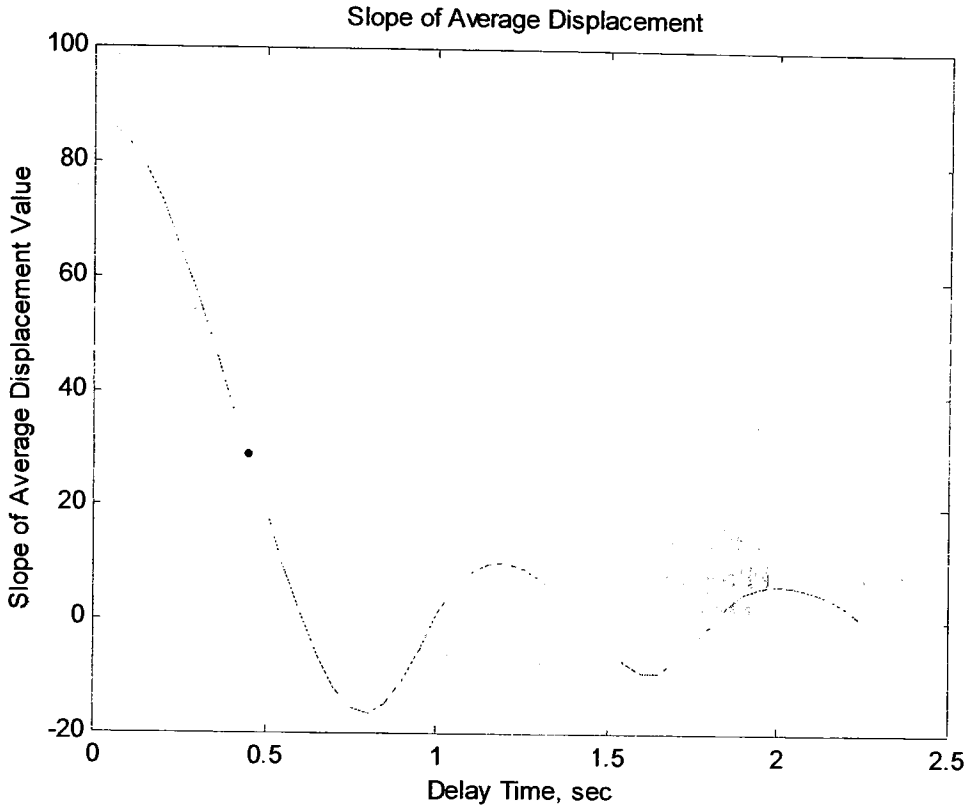
Delay Time value calculation program using average displacement by Andrew Dick
30-Jul-2003
13:16:4.036

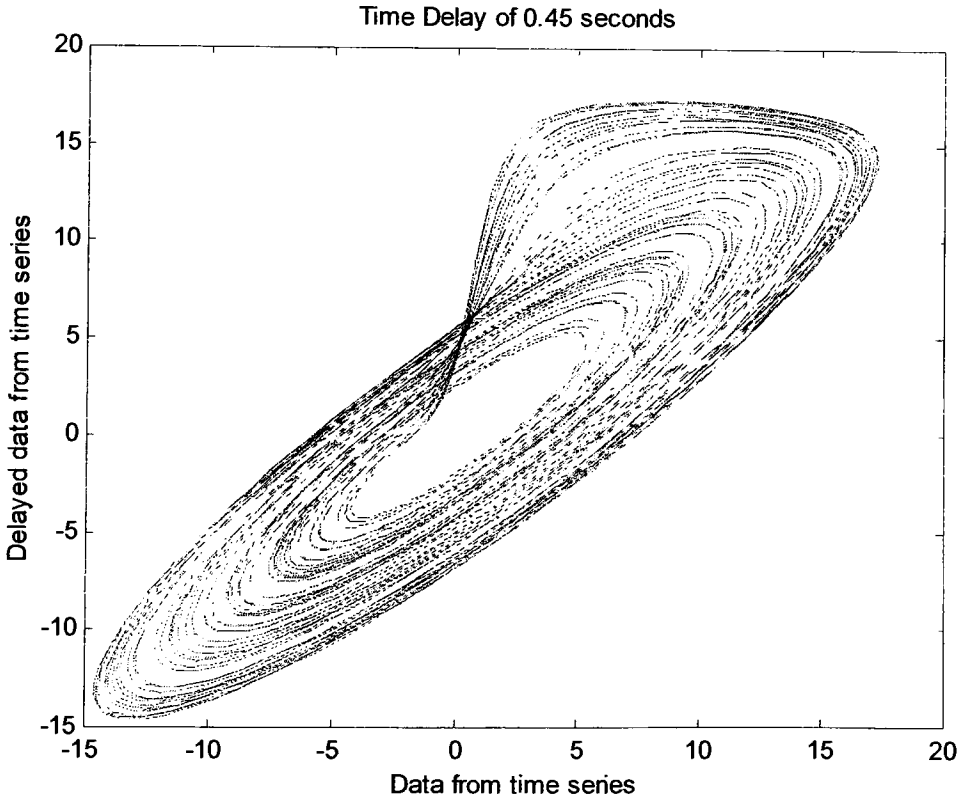
Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Data set consisted of 5000 data points with a time step of 0.05 seconds
A transient period of 40 seconds was removed

Delay Value of 0.45 seconds at 40% of slope at S(1)

13:16:15.702
Elapsed time 0 minutes, 11.676 seconds
Normal Termination of Program AVERAGE_DISPLACEMENT.M
****




Slope of Average Displacement
Slope of Average Displacement Value
Delay Time, sec
100
80
60
40
20
0
-20
0
0.5
1
1.5
2
2.5


Time Delay of 0.45 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20

****

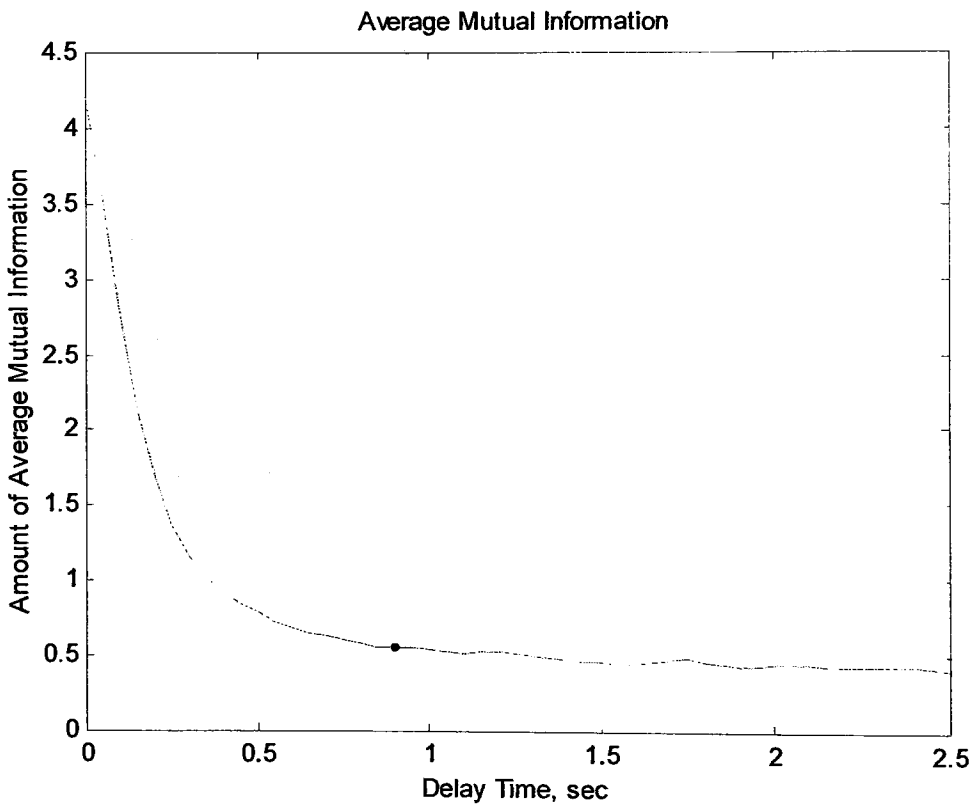
Delay Time value calculation program using Average Mutual Information by Andrew Dick
28-Jul-2003
14:10:7.966

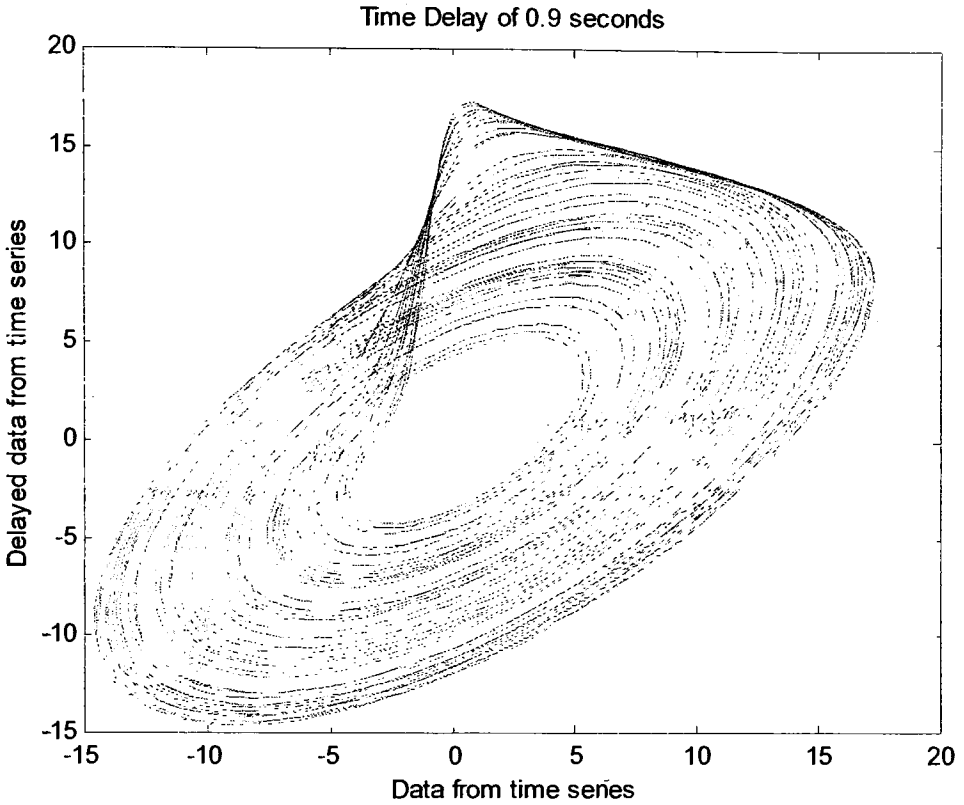
Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Data set consisted of 5000 data points with a time step of 0.05 seconds
A transient period of 40 seconds was removed

Delay Value for First Local Minimum is 0.9 seconds

14:14:18.216
Elapsed time 4 minutes, 10.25 seconds
Normal Termination of Program MUTUAL_INFORMATION.M

****



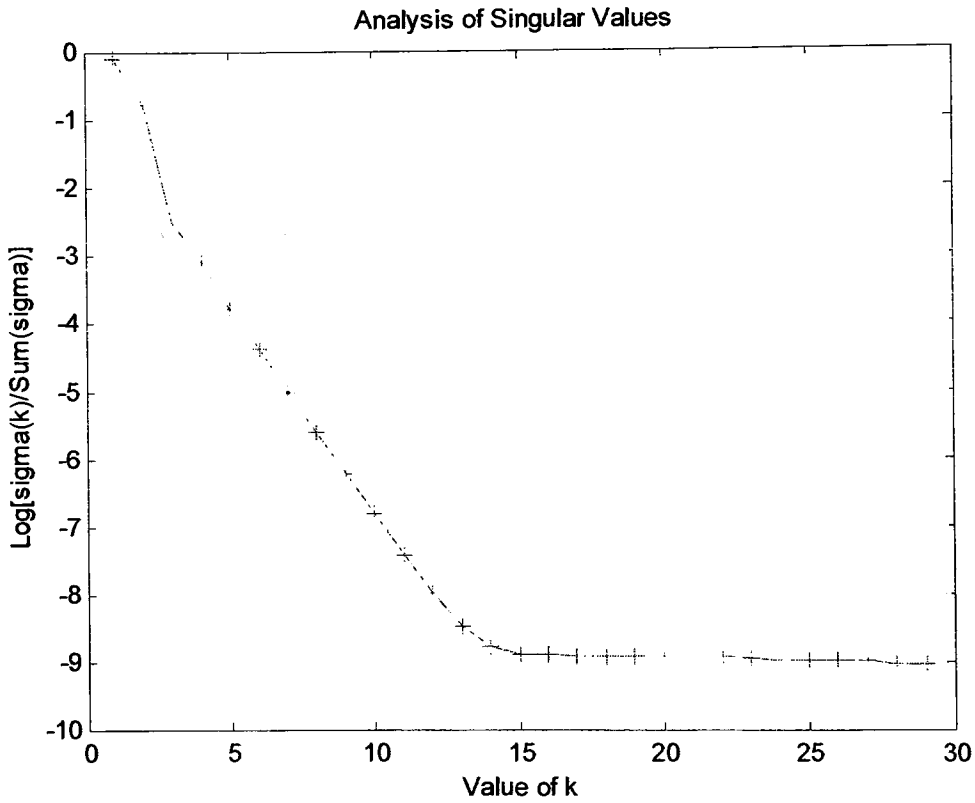

Time Delay of 0.9 seconds
Delayed data from time series
Data from time series
20
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20

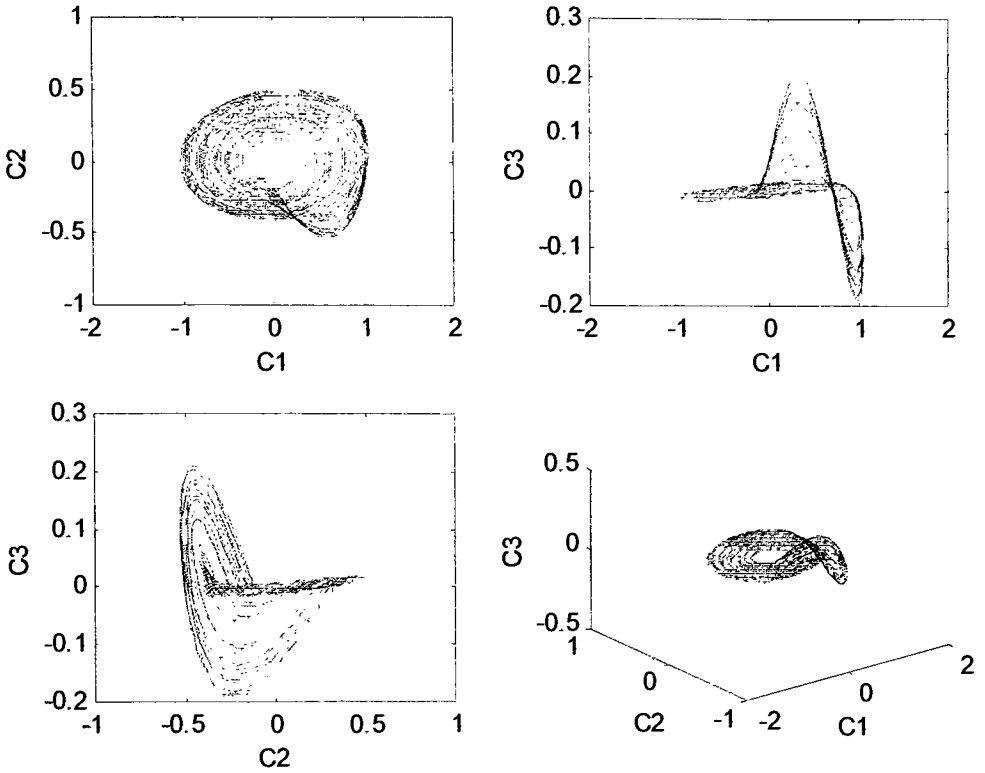
****

Singular System Approach for attractor reconstruction by Andrew Dick
28-Jul-2003
14:14:42

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
5000 data points used with time step of 0.05
40 seconds of transient data removed

14:15:3.872
Elapsed time 0 minutes, 21.872 seconds
Normal Termination of Program SINGULAR_SYSTEM_APPROACH.M
****




C2
C1
C3
C1
C3
C2
C3
C2
C1

APPENDIX C10

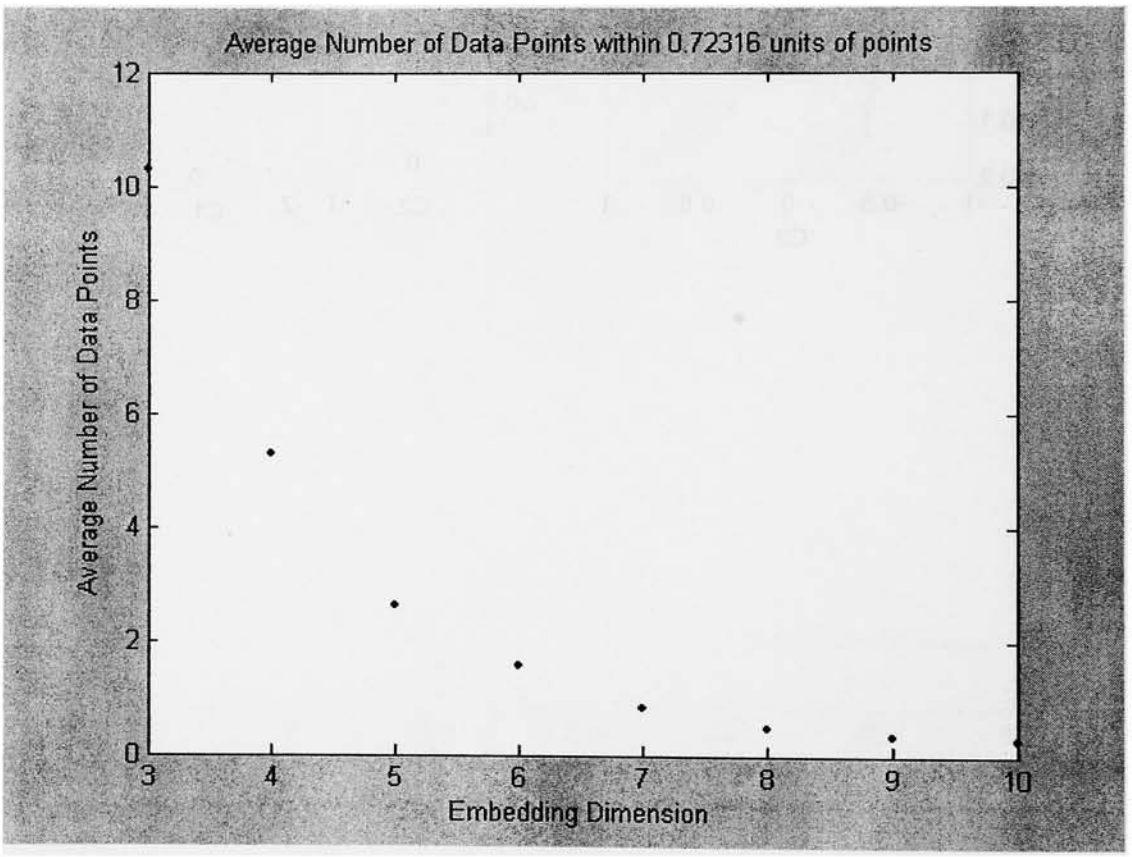
****

Calculation of Embedding Dimension by Andrew Dick
30-Jul-2003
9:28:9.375

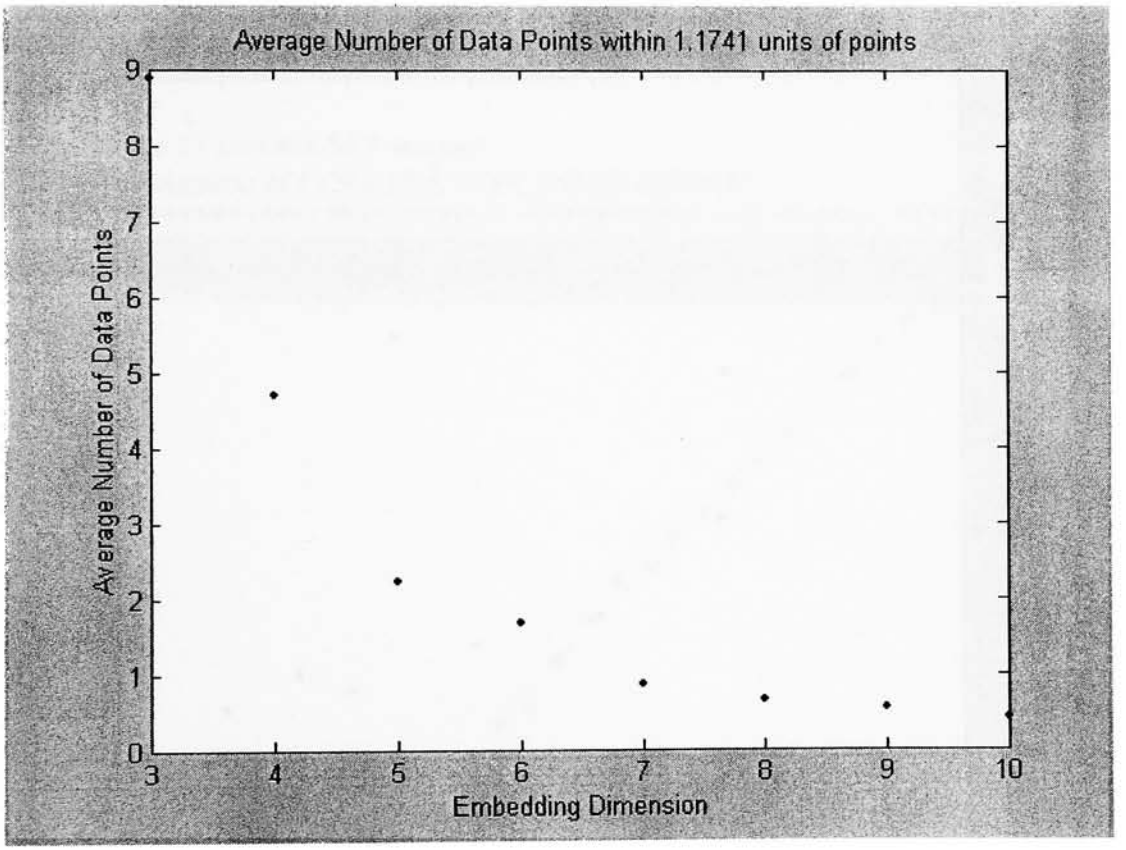
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
To remove a sufficient amount of false nearest neighbors,
an embedding dimension of 7 is required

9:45:15.218
Elapsed time 17 minutes, 5.859 seconds
Normal Termination of Program EMBEDDING_DIMENSION.M

****



```
************************************************************
Calculation of Embedding Dimension by Andrew Dick
30-Jul-2003
9:45:15.312

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, σ = 16, r = 45.92, b = 4
To remove a sufficient amount of false nearest neighbors,
  an embedding dimension of 7 is required

10:2:29.656
Elapsed time 17 minutes, 14.344 seconds
Normal Termination of Program EMBEDDING_DIMENSION.M
************************************************************
```



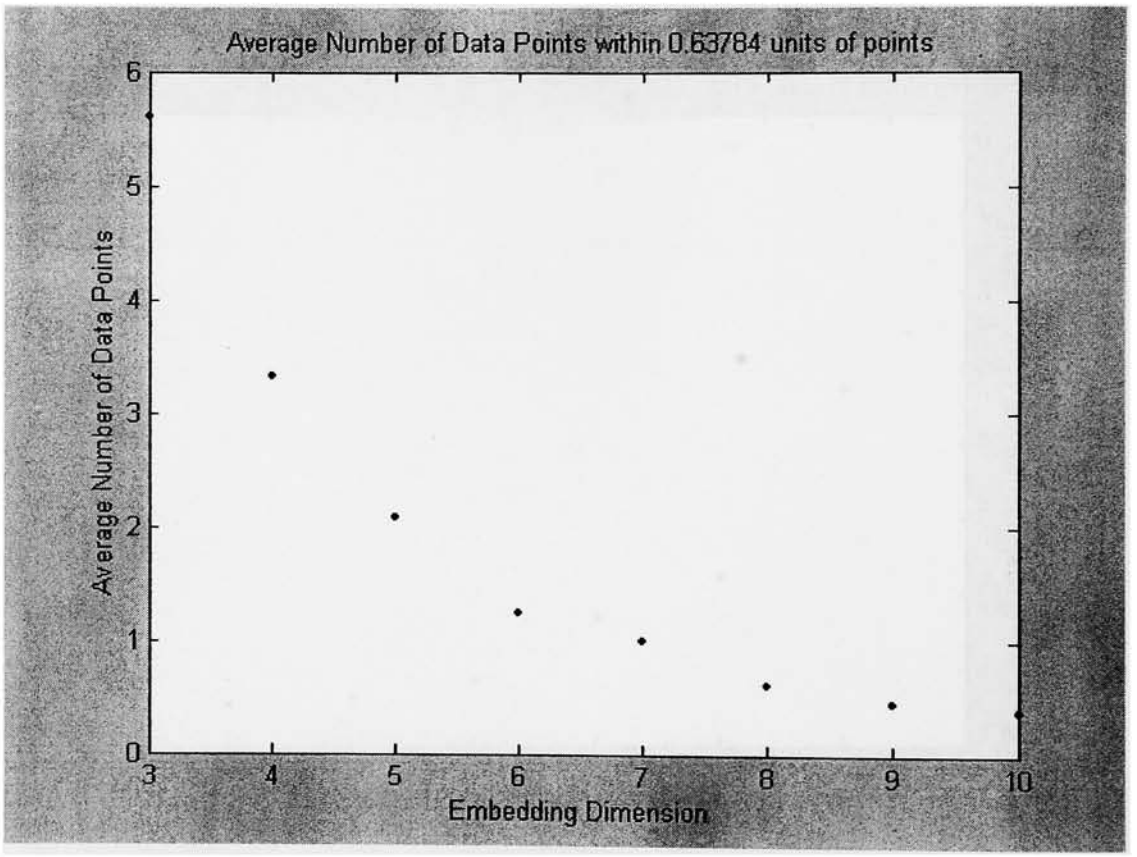
****

Calculation of Embedding Dimension by Andrew Dick
30-Jul-2003
10:2:29.828

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
To remove a sufficient amount of false nearest neighbors,
an embedding dimension of 8 is required

10:19:18.171
Elapsed time 16 minutes, 48.343 seconds
Normal Termination of Program EMBEDDING_DIMENSION.M

****



APPENDIX C11

****

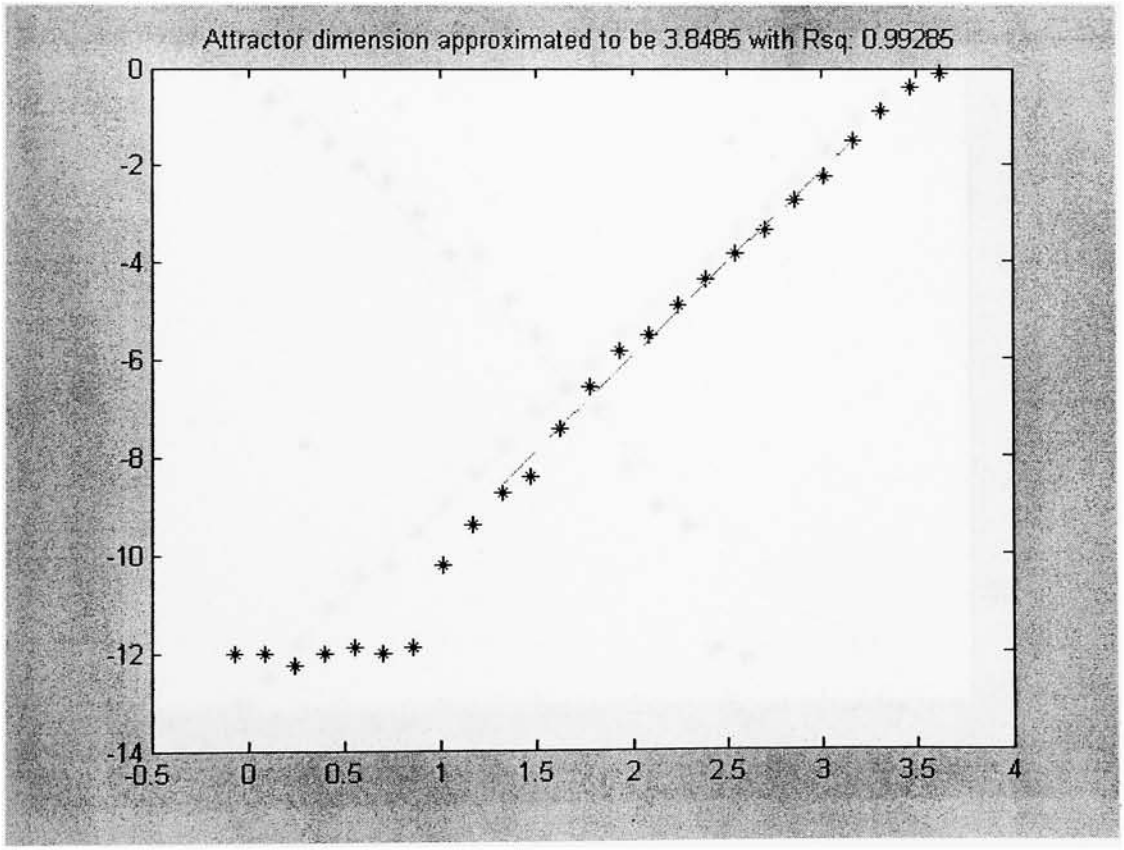
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
9:37:45.171

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using First Zero of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4316 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 3.8485

9:55:41.406
Elapsed time 17 minutes 56.5 seconds
Normal termination of CORRELATION_DIMENSION.M

****



****

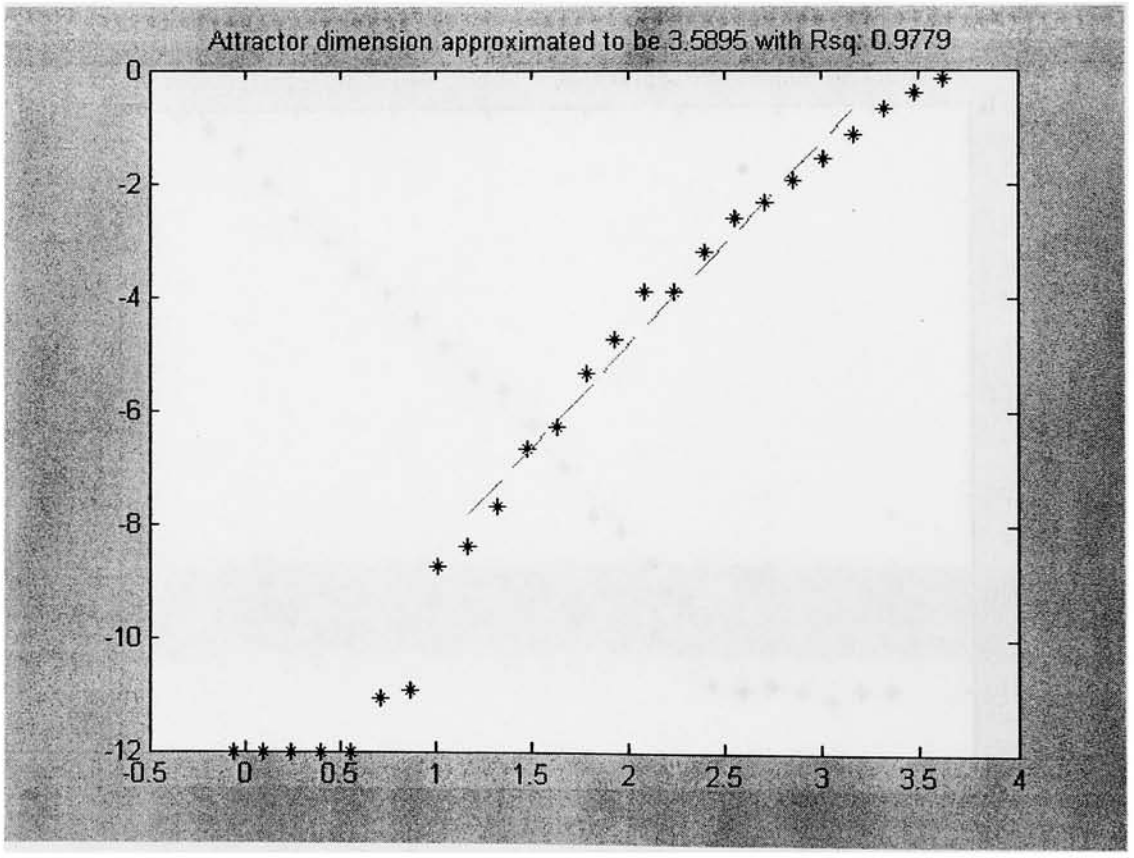
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
9:55:42.531

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using First Minimum of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4520 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 3.5895

10:14:27.906
Elapsed time 18 minutes 45.391 seconds
Normal termination of CORRELATION_DIMENSION.M

****



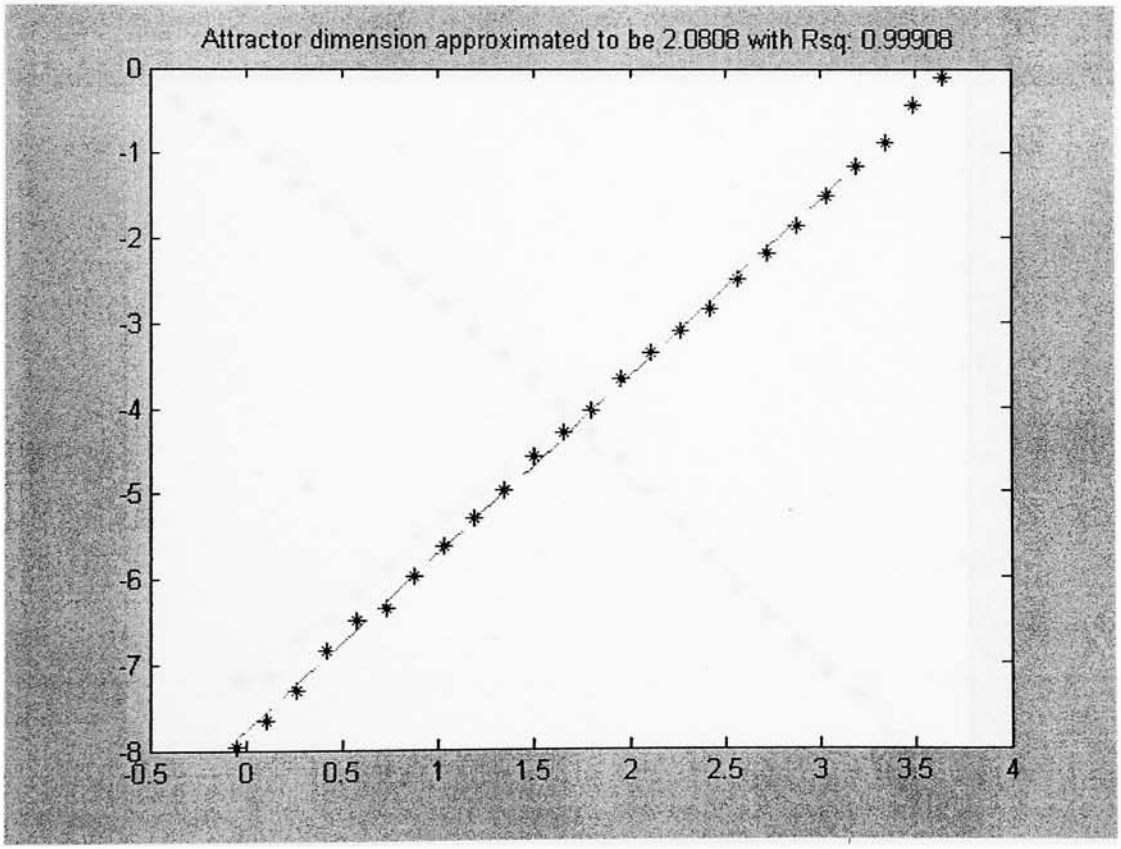
****

Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
10:14:29.515

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using First Inflection Point of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4904 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 2.0808

10:34:52.531
Elapsed time 20 minutes 23.094 seconds
Normal termination of CORRELATION_DIMENSION.M
****



****

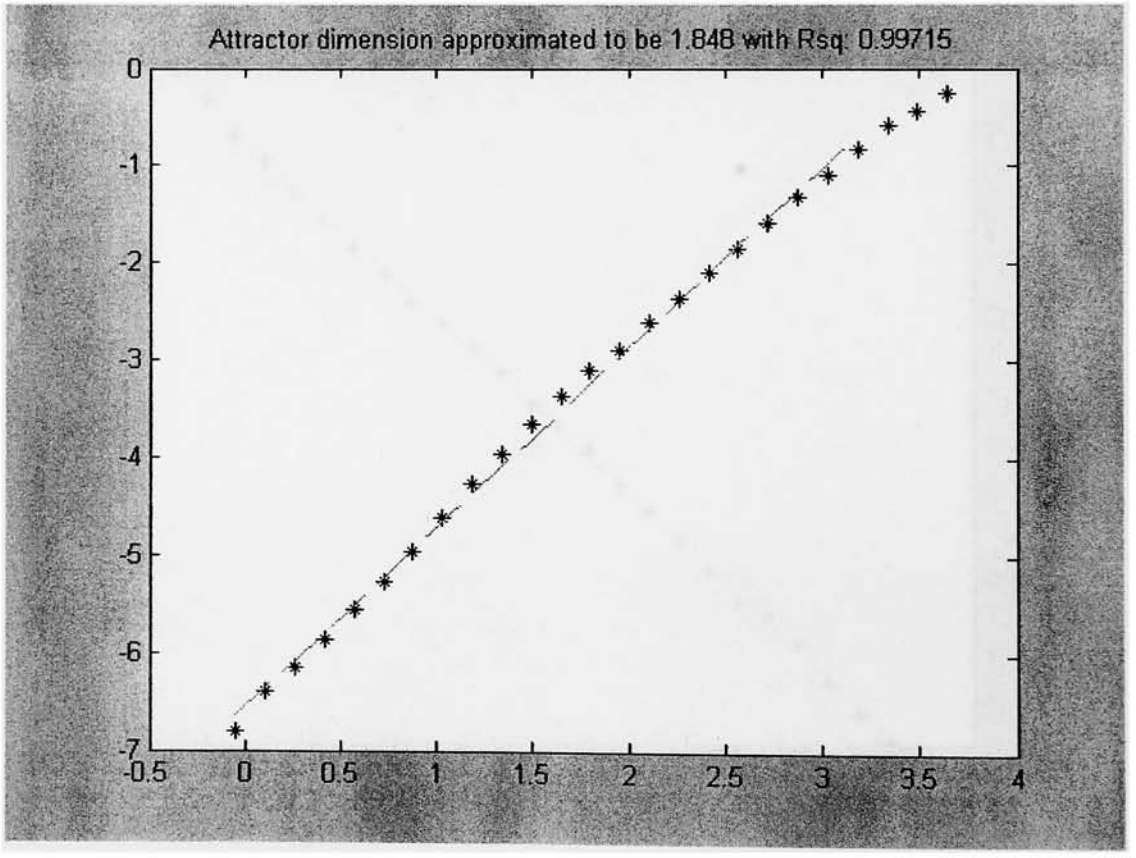
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
10:55:27.89

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using Average Displacement Method
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4958 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 1.848

11:16:6.093
Elapsed time 20 minutes 38.25 seconds
Normal termination of CORRELATION_DIMENSION.M

****



****

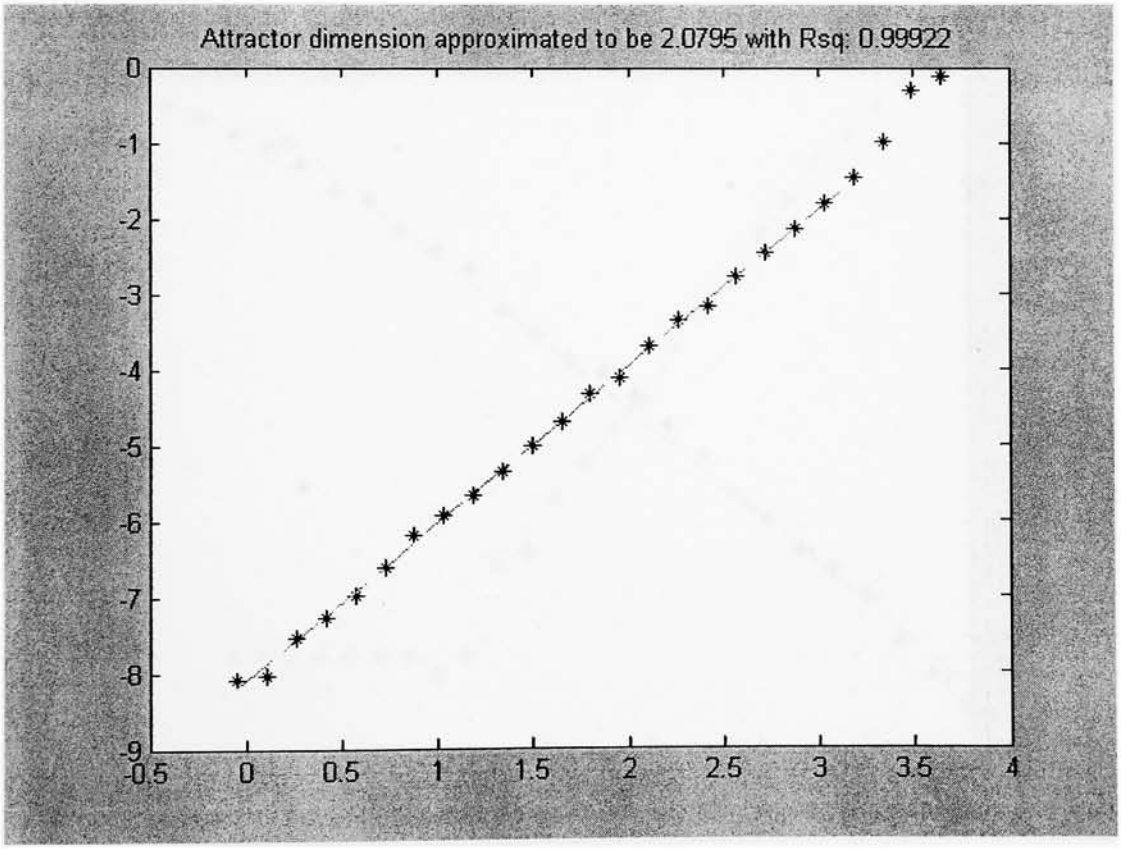
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
11:16:6.718

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using Average Mutual Information Method
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4958 data points
Calculated using a Theiler coefficient of W=10

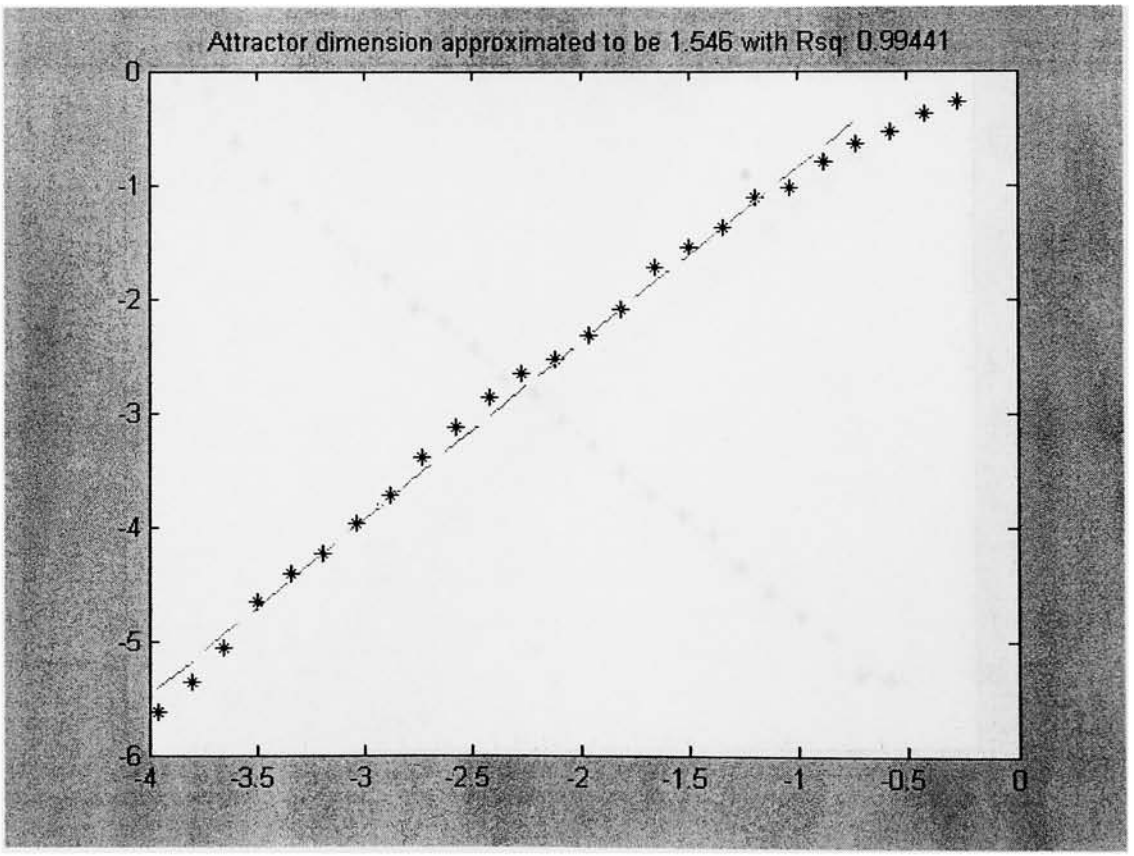
The Correlation Dimension of the data set was determined to be 2.0795

11:36:41.781
Elapsed time 20 minutes 35.11 seconds
Normal termination of CORRELATION_DIMENSION.M

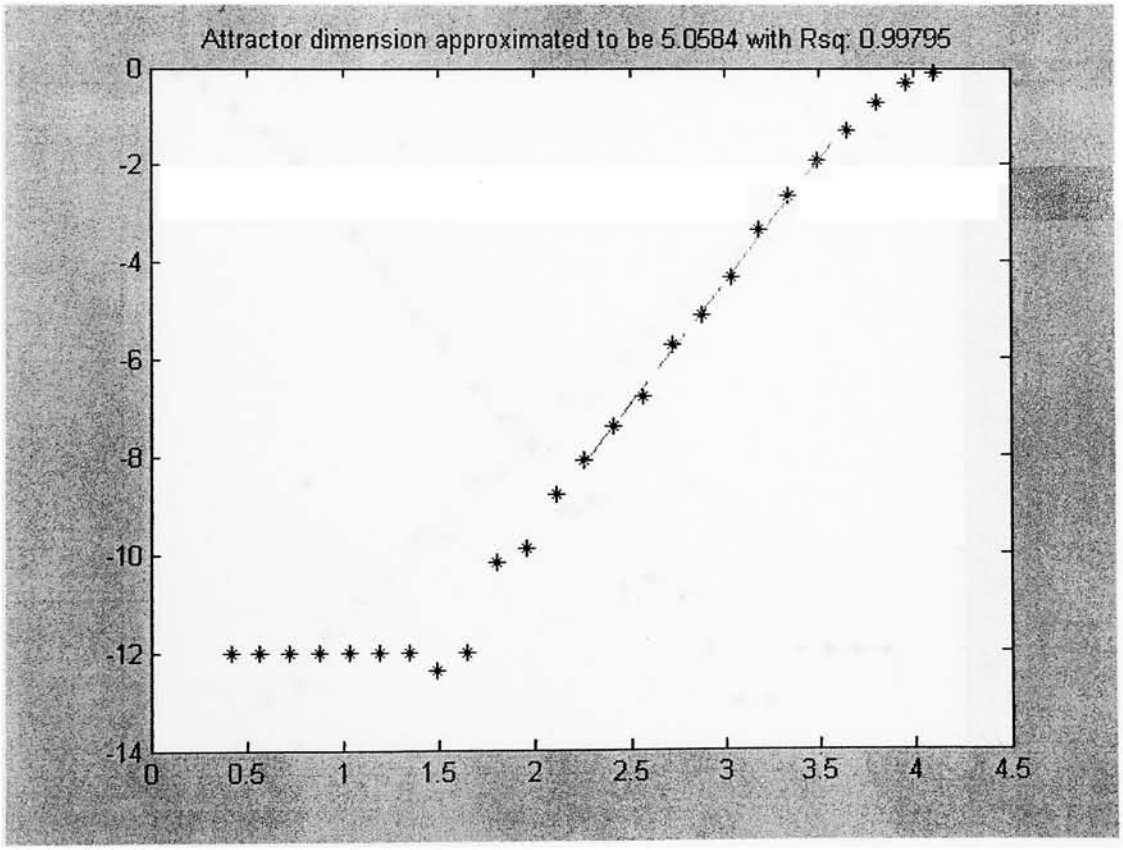
****



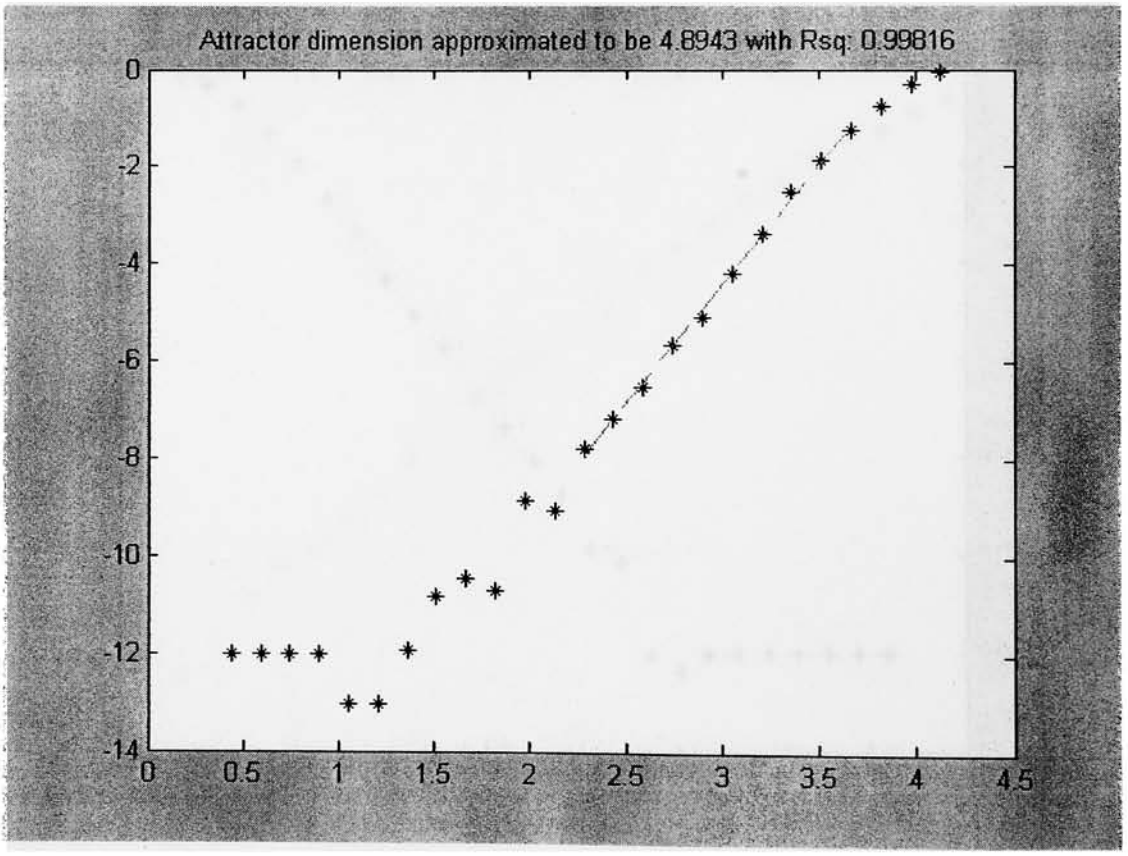
```
**********************************************************************
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
11:36:43.421

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, σ = 10, r = 28, b = 8/3
Reconstructed using Singular System Approach
   and embedding dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4985 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 1.546

11:57:28.875
Elapsed time 20 minutes 45.454 seconds
Normal termination of CORRELATION_DIMENSION.M
**********************************************************************
```



```
**********************************************************************
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
7:27:21.64

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, σ = 16, r = 45.92, b = 4
Reconstructed using First Zero of Autocorrelation
   and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 2372 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 5.0584

7:37:8.875
Elapsed time 9 minutes 47.5 seconds
Normal termination of CORRELATION_DIMENSION.M
**********************************************************************
```



```
**********************************************************************
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
7:37:13.046

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, σ = 16, r = 45.92, b = 4
Reconstructed using First Minimum of Autocorrelation
   and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4550 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 4.8943

7:56:3.015
Elapsed time 18 minutes 50 seconds
Normal termination of CORRELATION_DIMENSION.M
**********************************************************************
```

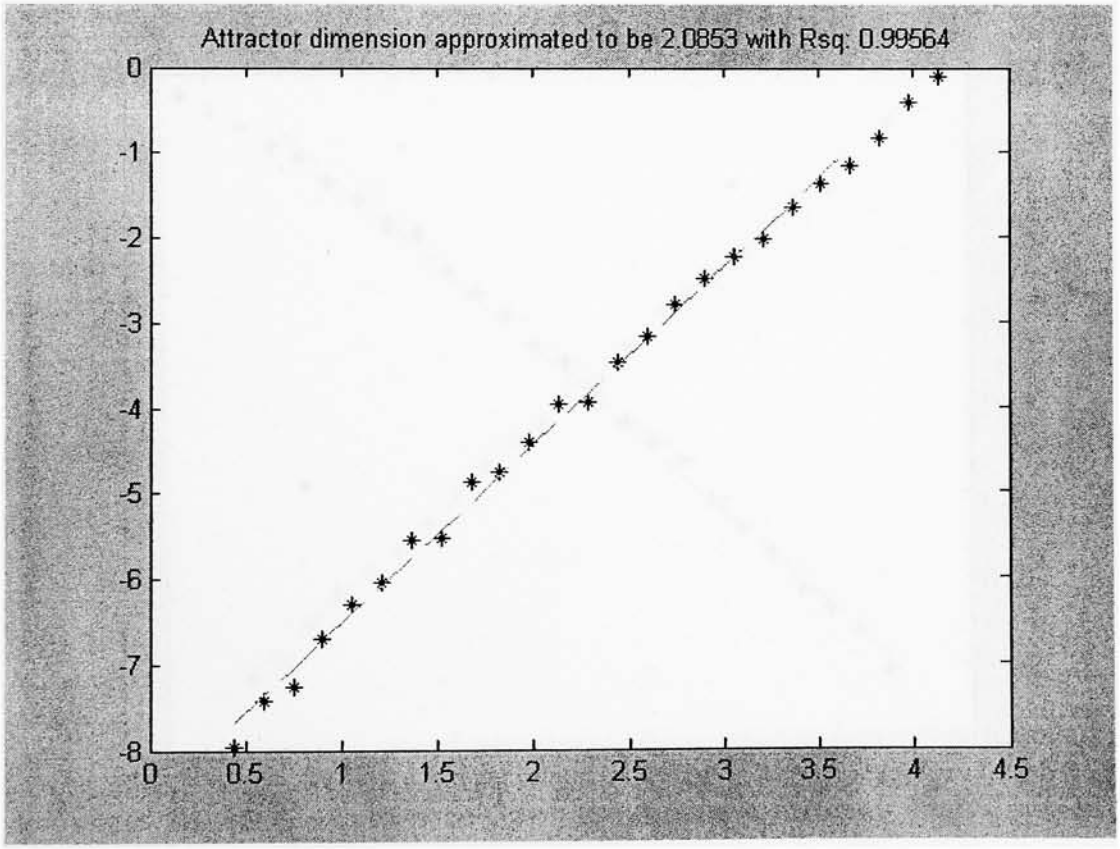


****
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
7:56:4.625

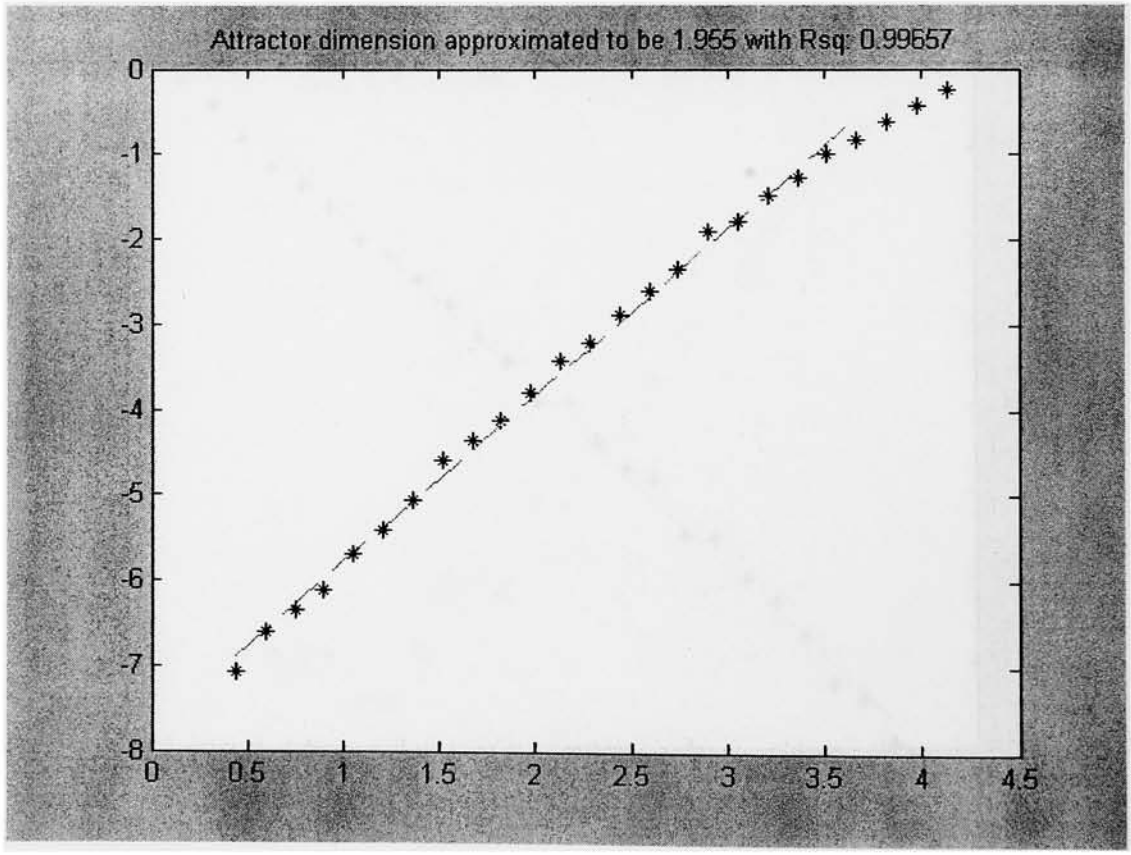
Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using First Inflection Point of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4940 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 2.0853

8:16:33.234
Elapsed time 20 minutes 28.656 seconds
Normal termination of CORRELATION_DIMENSION.M
****



```
**********************************************************************
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
8:37:8.031

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, σ = 16, r = 45.92, b = 4
Reconstructed using Average Displacement Method
   and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4970 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 1.955

8:57:46.109
Elapsed time 20 minutes 38.109 seconds
Normal termination of CORRELATION_DIMENSION.M
**********************************************************************
```



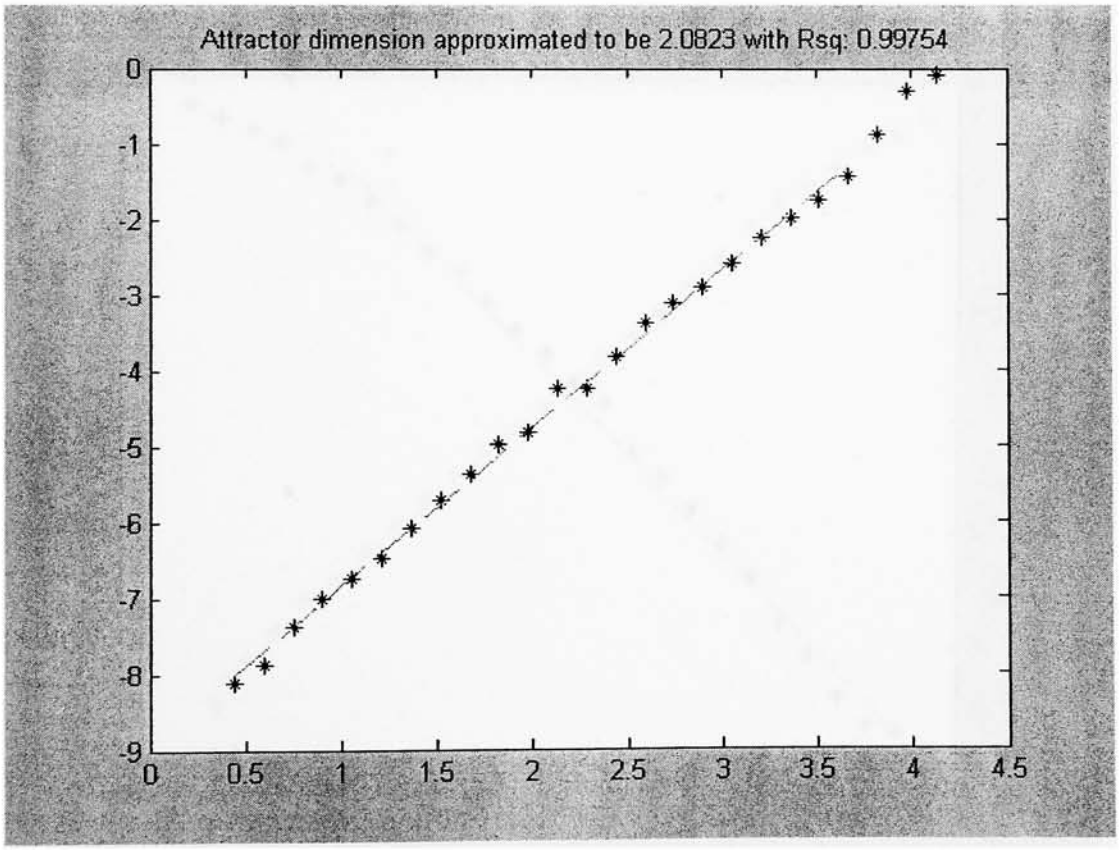
****

Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
8:57:46.718

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using Average Mutual Information Method
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4970 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 2.0823

9:18:23.515
Elapsed time 20 minutes 36.828 seconds
Normal termination of CORRELATION_DIMENSION.M
****



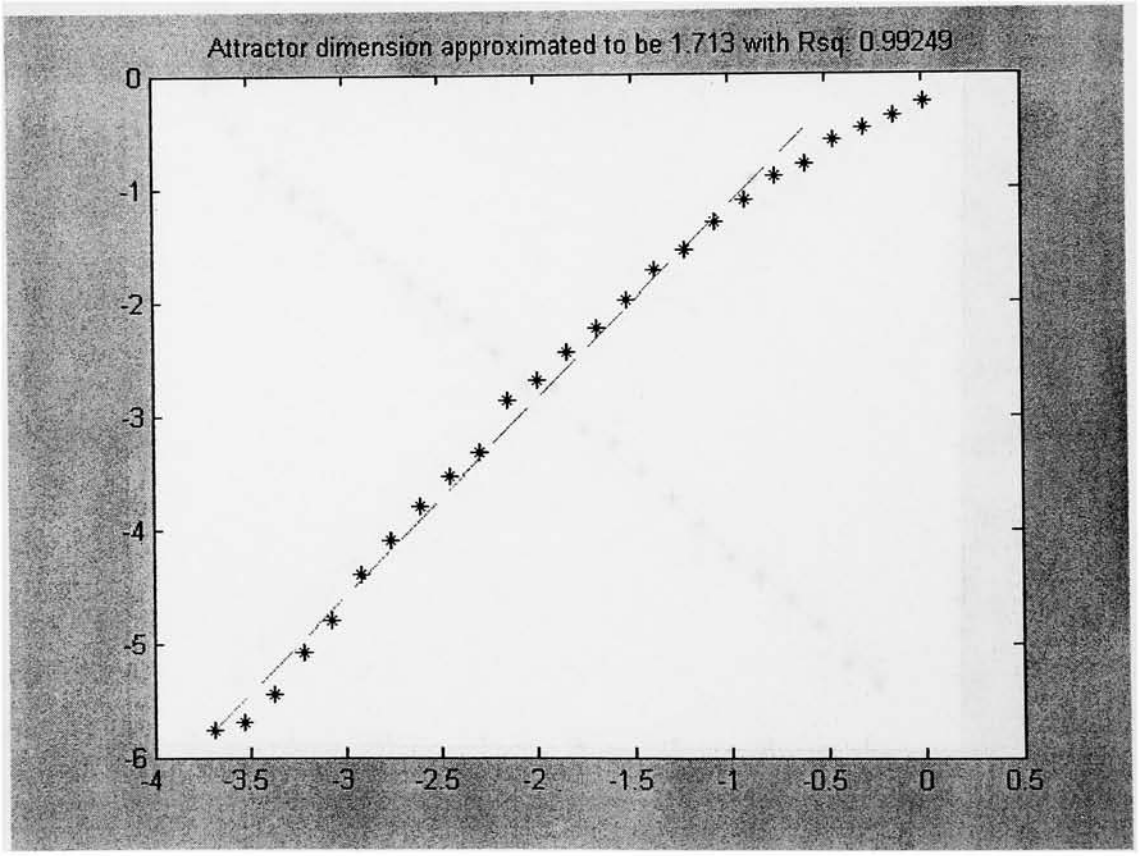
****

Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
9:18:25.031

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using Singular System Approach
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated using 4991 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 1.713

9:39:9.171
Elapsed time 20 minutes 44.156 seconds
Normal termination of CORRELATION_DIMENSION.M
****



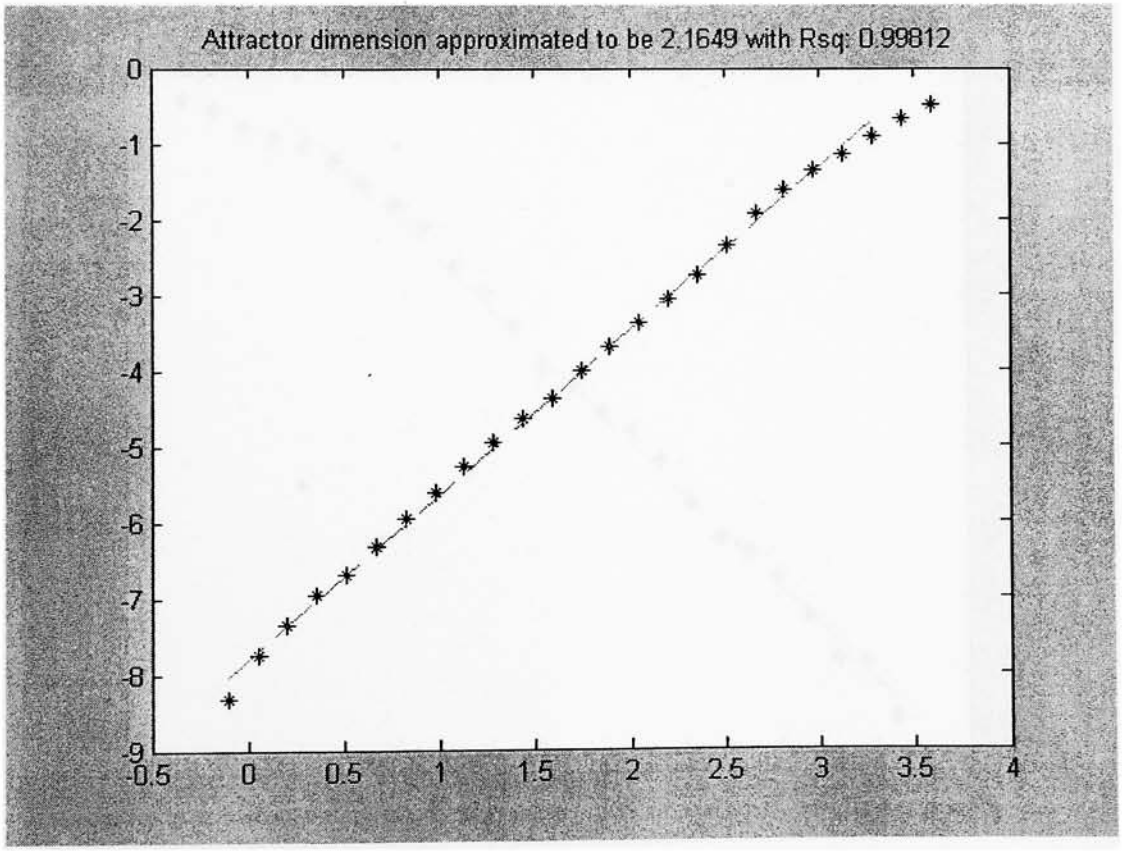
****

Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
10:18:32.265

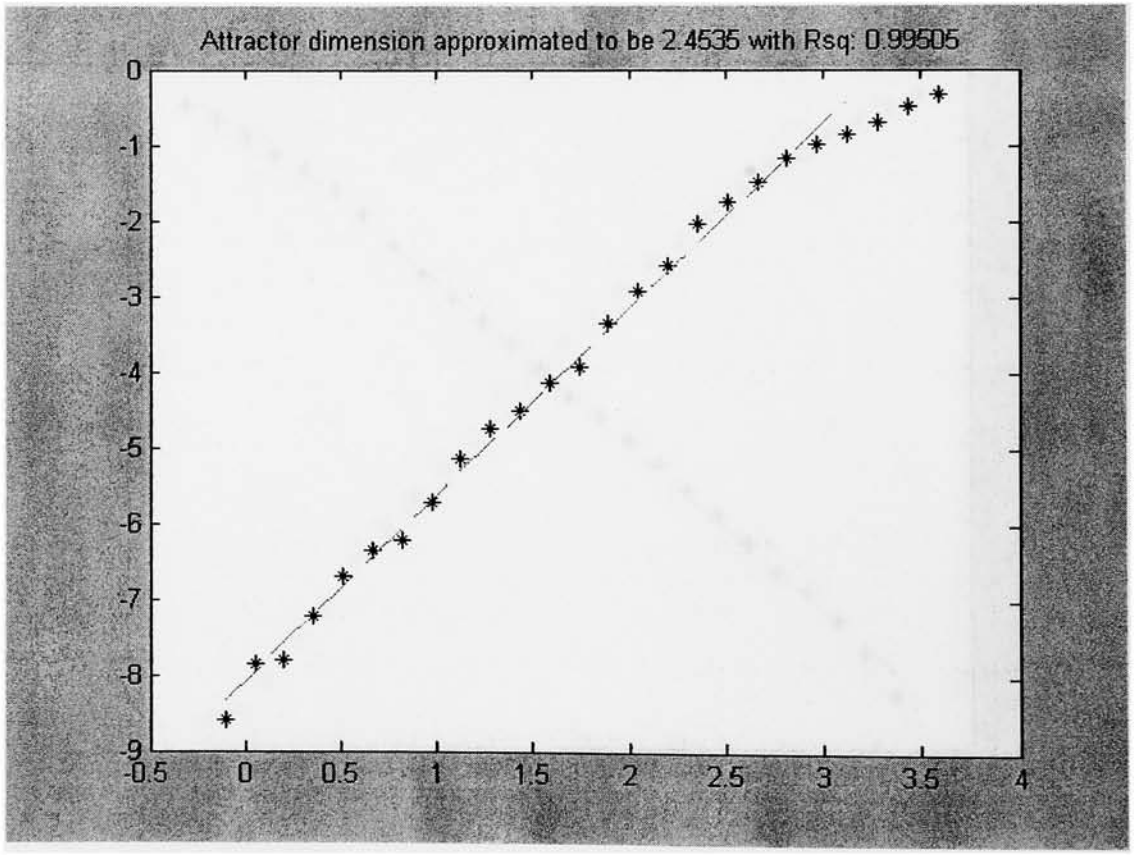
Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using First Zero of Autocorrelation
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated using 4790 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 2.1649

10:40:28.796
Elapsed time 21 minutes 56.875 seconds
Normal termination of CORRELATION_DIMENSION.M
****



```
**********************************************************************
Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
10:40:29.531

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, a = 0.15, b = 0.20, c = 10
Reconstructed using First Minimum of Autocorrelation
   and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated using 4790 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 2.4535

11:2:27.109
Elapsed time 21 minutes 57.641 seconds
Normal termination of CORRELATION_DIMENSION.M
**********************************************************************
```



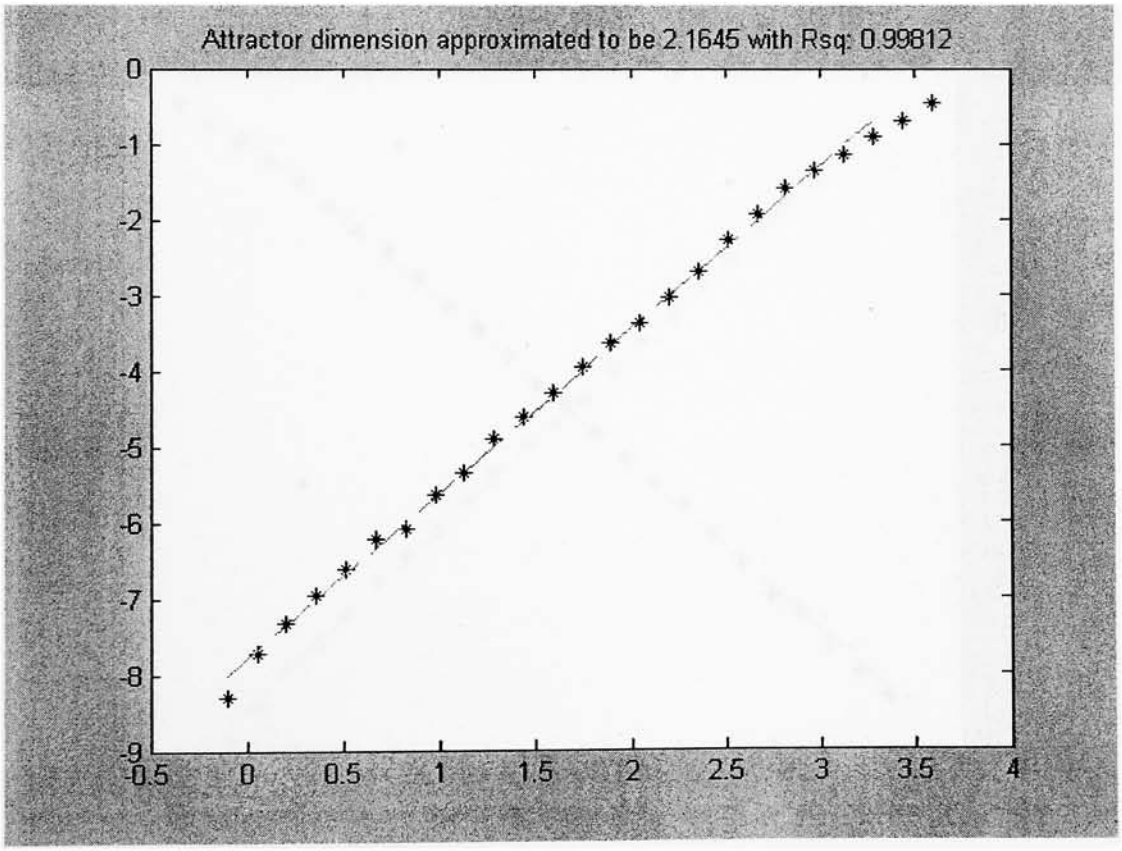
****

Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
11:2:27.859

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using First Inflection Point of Autocorrelation
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated using 4804 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 2.1645

11:24:27.875
Elapsed time 22 minutes 0.094 seconds
Normal termination of CORRELATION_DIMENSION.M
****



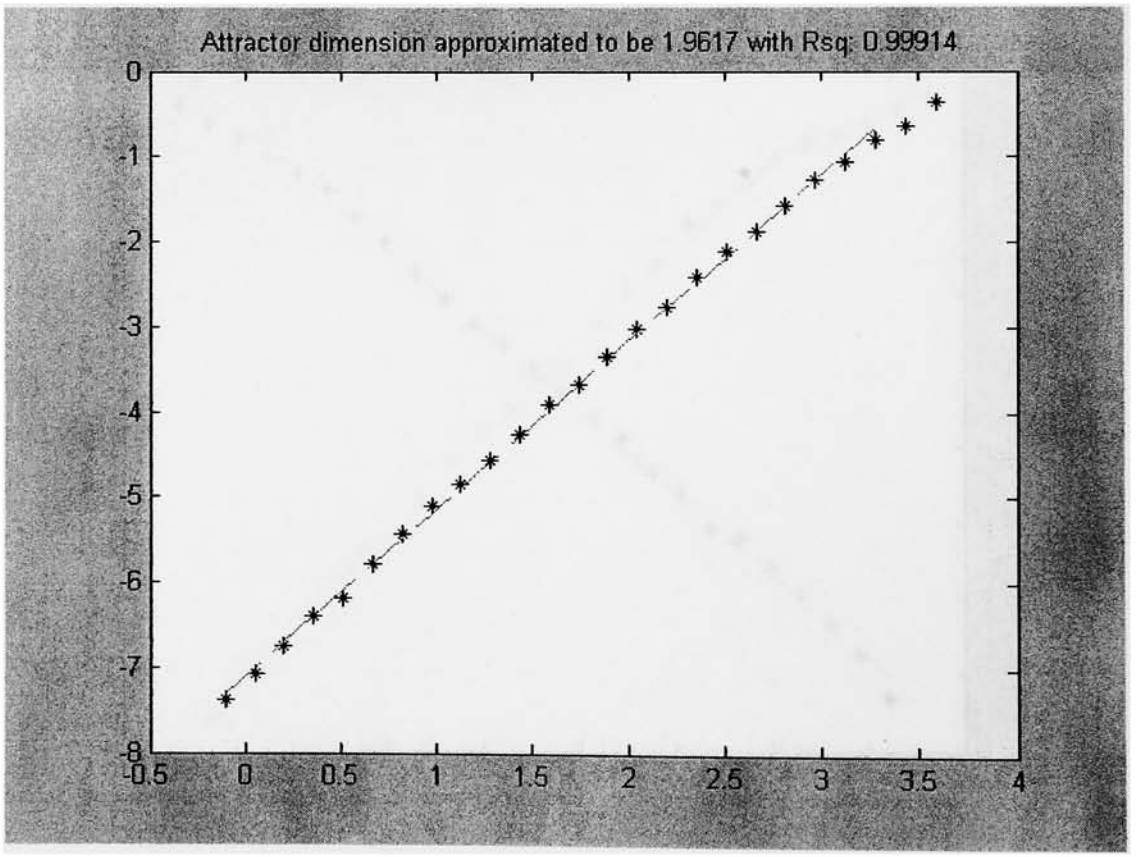
****

Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
11:46:39.609

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using Average Displacement Method
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated using 4937 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 1.9617

12:9:16.109
Elapsed time 22 minutes 36.547 seconds
Normal termination of CORRELATION_DIMENSION.M
****



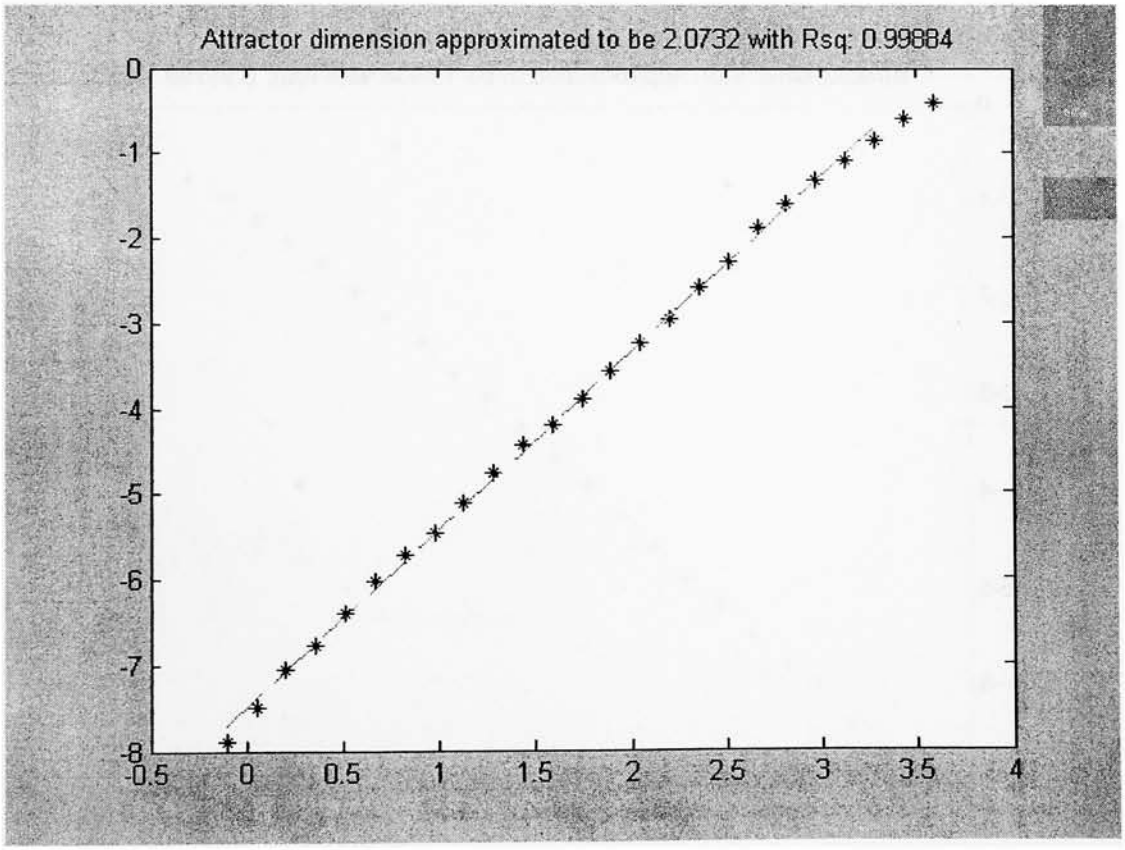
****

Correlation Dimension calculation program by Andrew Dick
12-Aug-2003
12:9:16.796

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using Average Mutual Information Method
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated using 4937 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 2.0732

12:31:52.359
Elapsed time 22 minutes 35.609 seconds
Normal termination of CORRELATION_DIMENSION.M
****



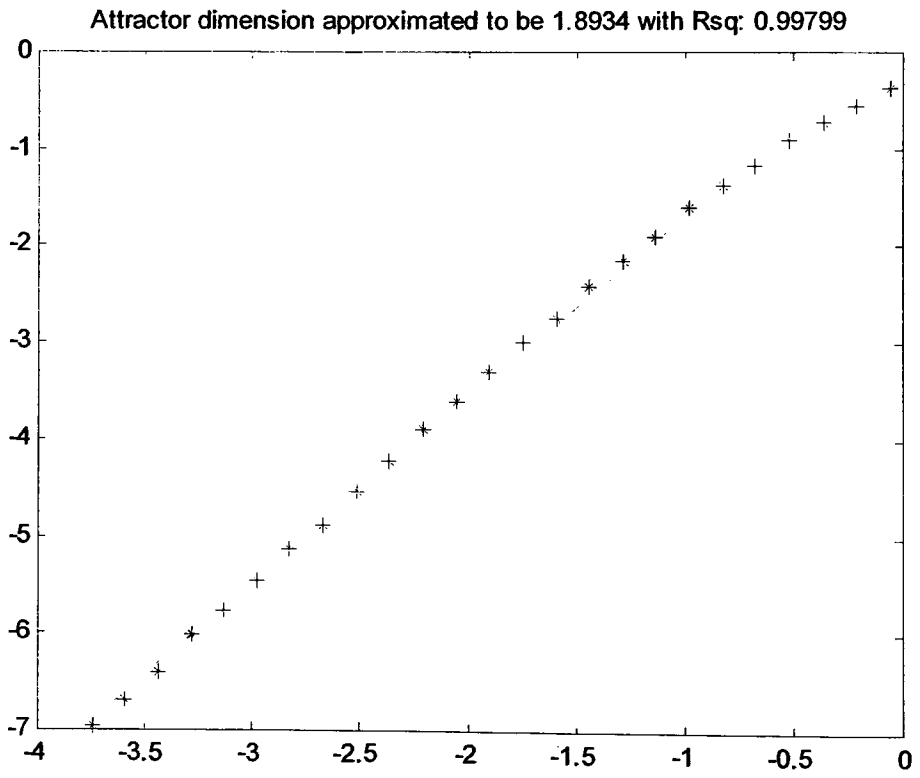
****

Correlation Dimension calculation program by Andrew Dick
14-Aug-2003
10:15:44.06

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using Singular System Approach
and Embedding Dimension of 13
First 0.05 seconds of transient trajectory removed
Calculated using 4985 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 1.8934

10:30:1.844
Elapsed time 14 minutes 17.794 seconds
Normal termination of CORRELATION_DIMENSION.M
****



APPENDIX C12

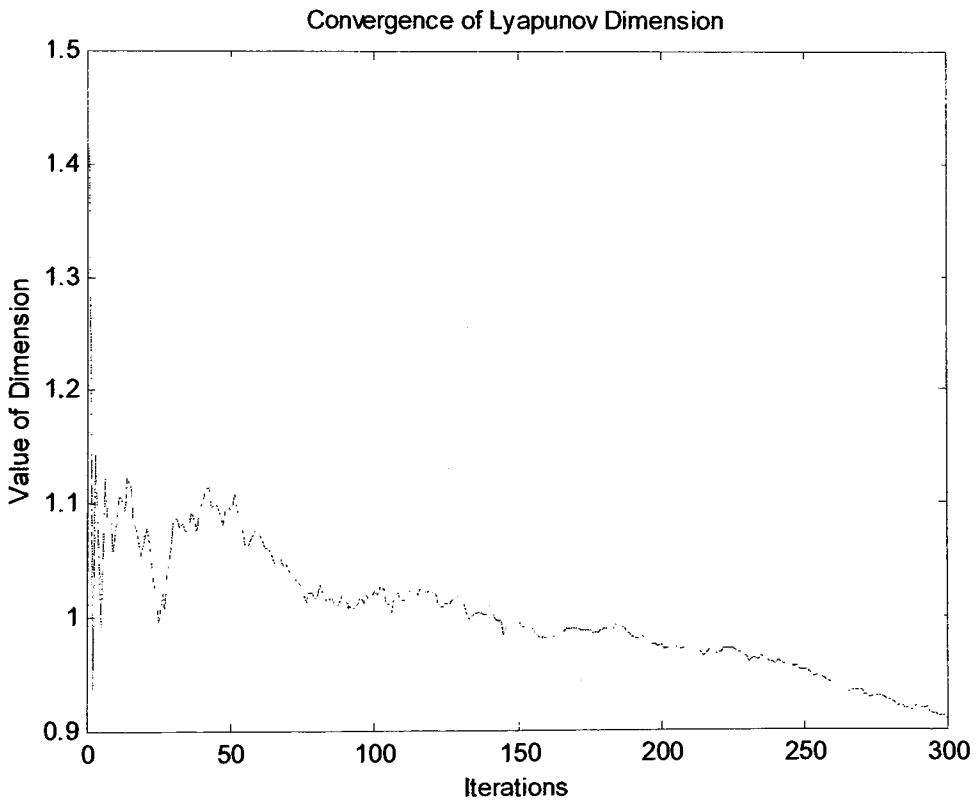
****

Largest Lyapunov exponent calculation program by Andrew Dick
19-Aug-2003
8:53:13.411

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using First Zero of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89316 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.90949

12:3:22.286
Elapsed time 190 minutes 8.945 seconds
Normal Termination of LARGEST_LYAPUNOV.M

****

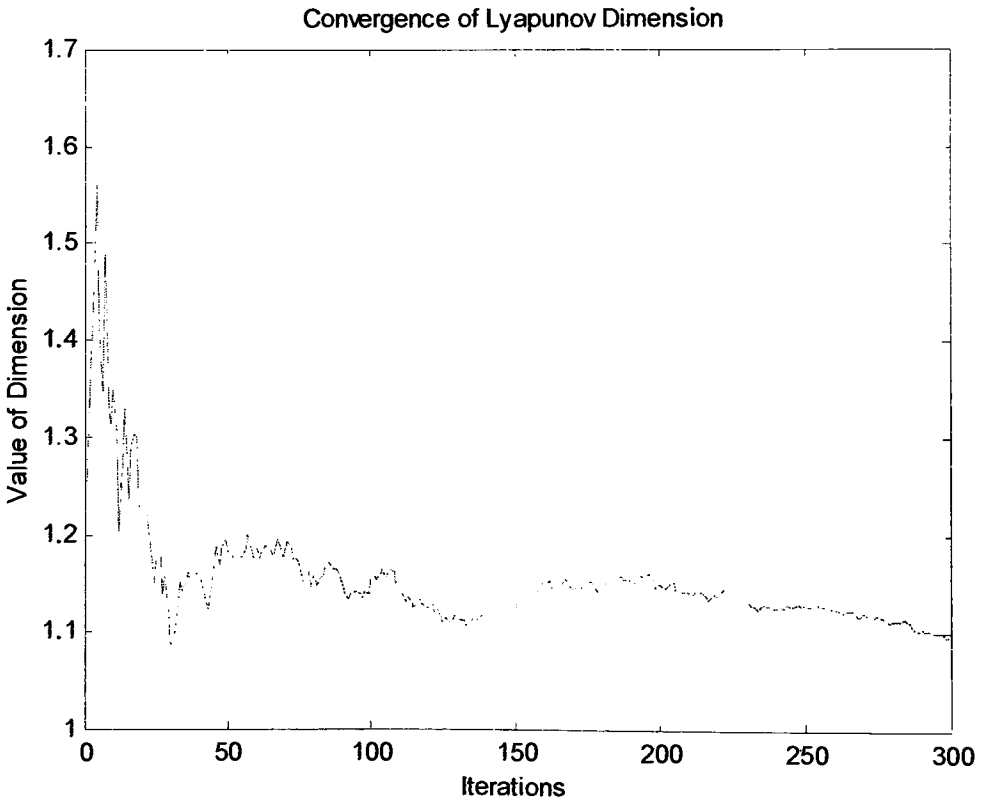


****

Largest Lyapunov exponent calculation program by Andrew Dick
19-Aug-2003
12:3:30.207

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using First Minimum of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89520 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 1.0963

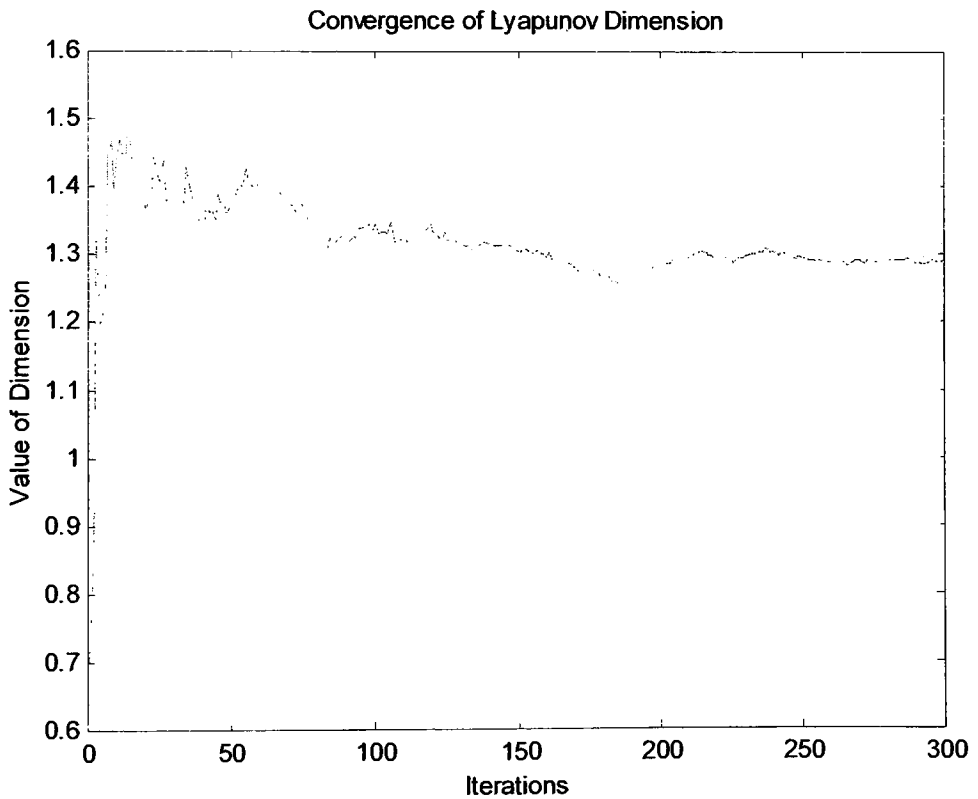
14:52:36.177
Elapsed time 169 minutes 5.97 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



****
Largest Lyapunov exponent calculation program by Andrew Dick
19-Aug-2003
14:52:47.853

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using First Inflection Point of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89904 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 1.2907

17:49:0.236
Elapsed time 176 minutes 12.383 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****

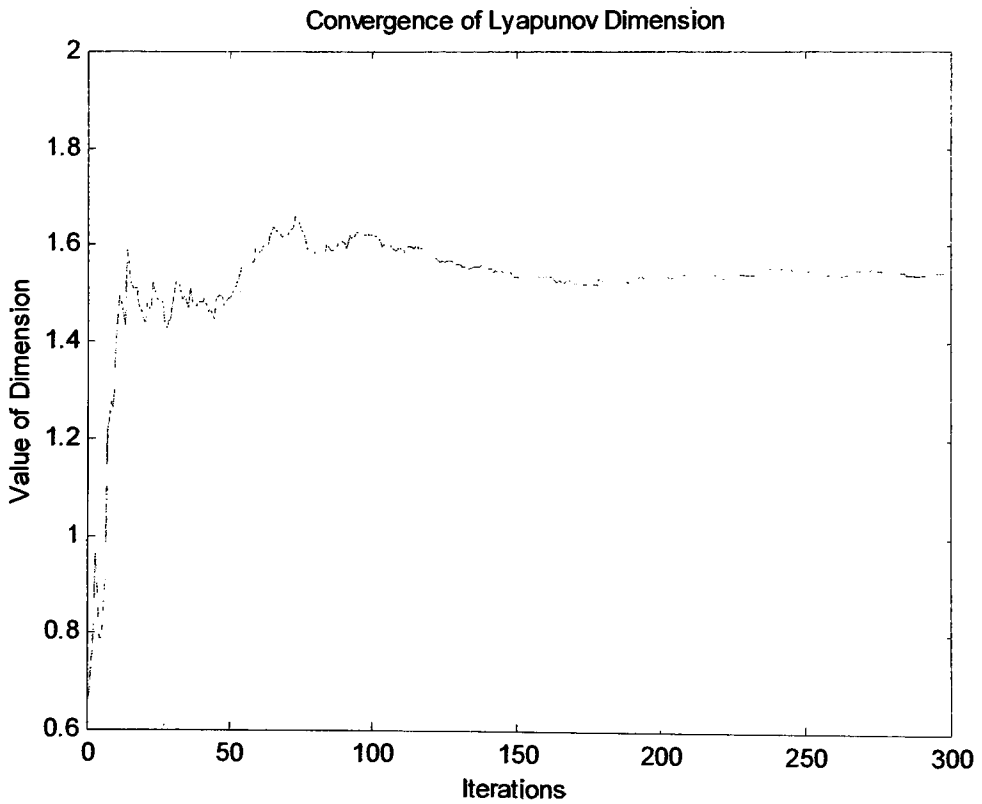


****

Largest Lyapunov exponent calculation program by Andrew Dick
19-Aug-2003
20:38:57.438

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using Average Displacement Method
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89958 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 1.5435

23:43:36.86
Elapsed time 184 minutes 39.422 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****

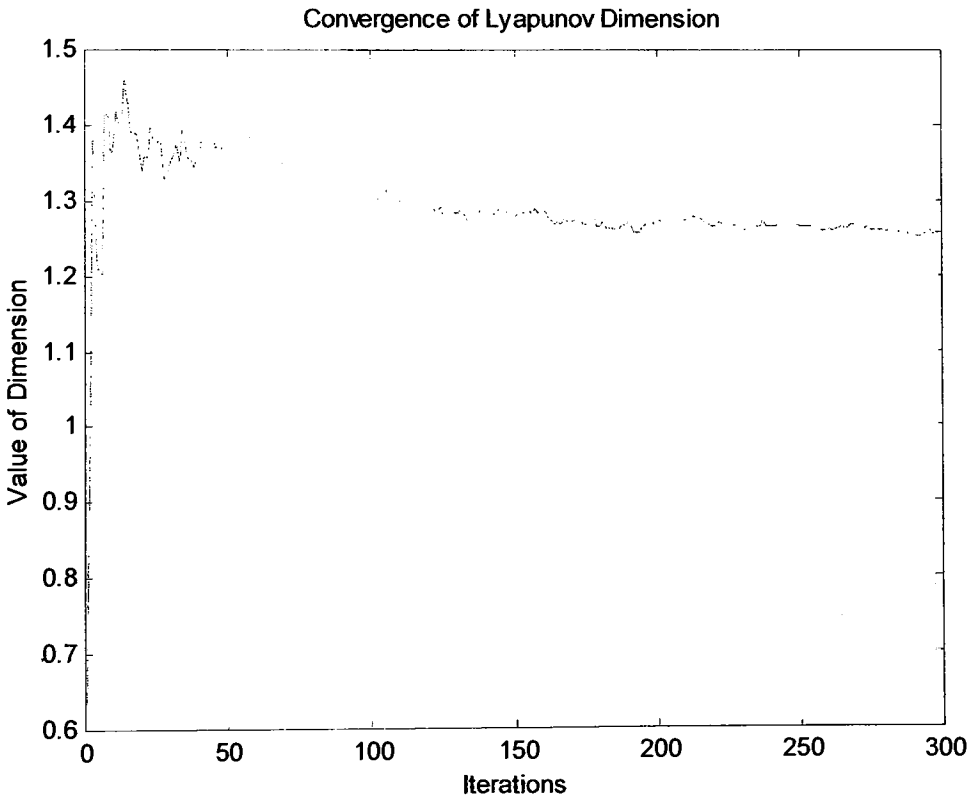


****

Largest Lyapunov exponent calculation program by Andrew Dick
19-Aug-2003
23:43:39.584

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using Average Mutual Information Method
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89958 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 1.2562

2:45:29.512
Elapsed time 181 minutes 49.928 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****

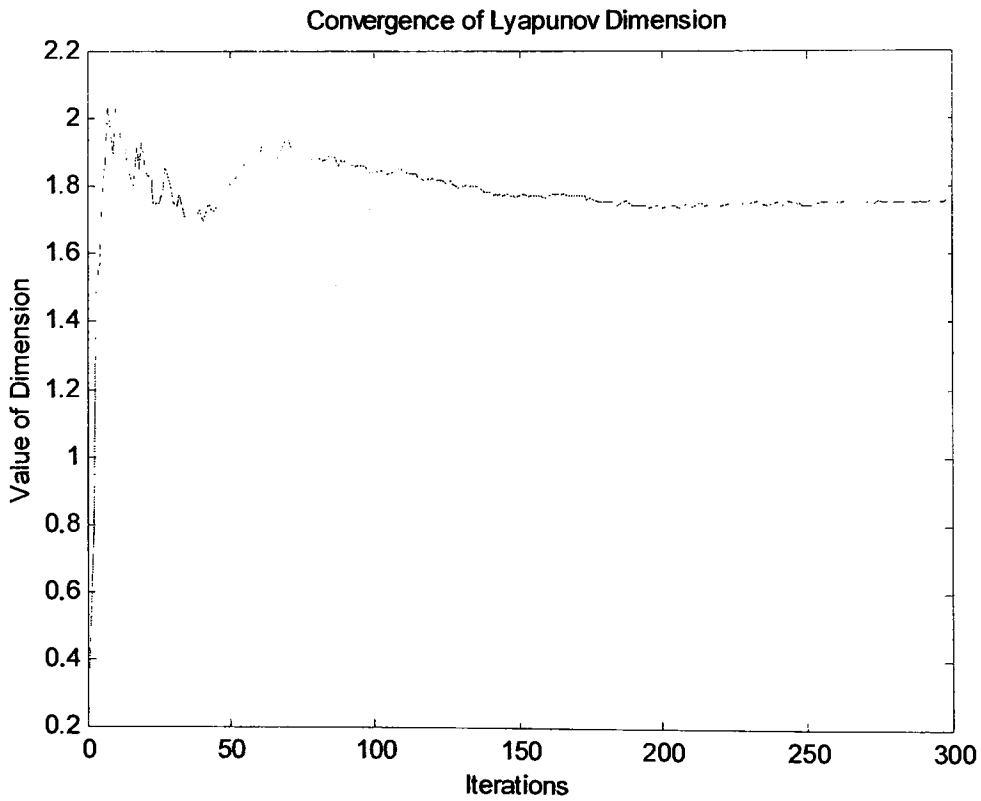


****

Largest Lyapunov exponent calculation program by Andrew Dick
26-Aug-2003
18:1:21.013

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Classical Parameter Set, $\sigma = 10$, $r = 28$, $b = 8/3$
Reconstructed using Singular System Approach
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89985 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 1.7565

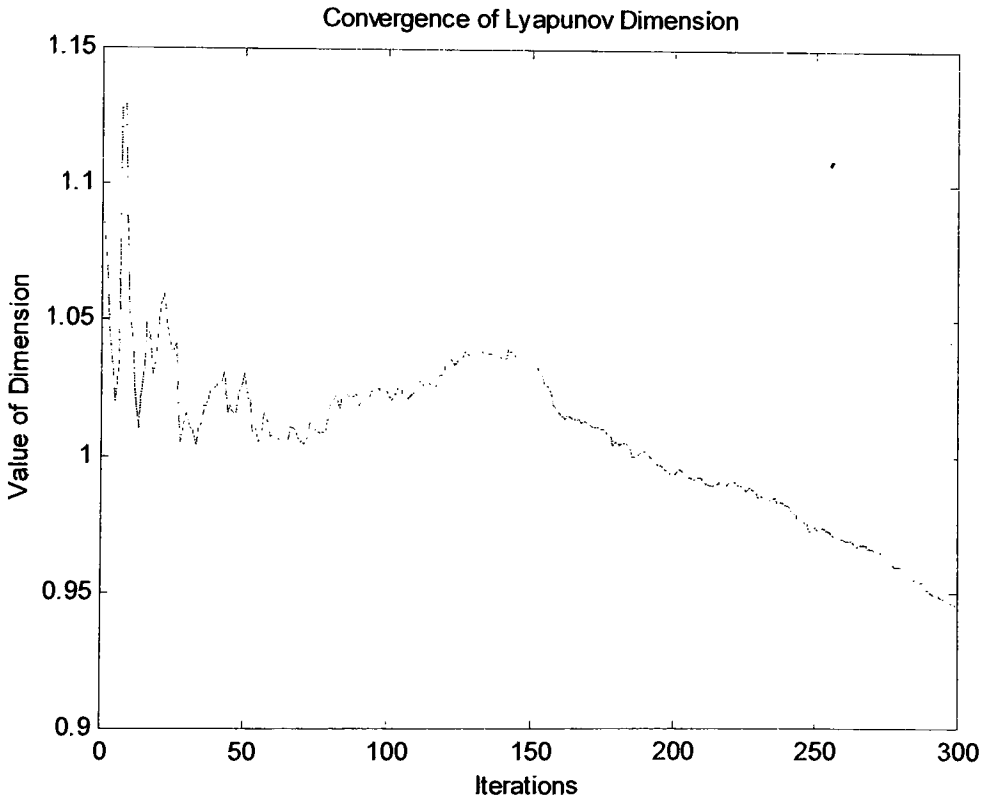
18:12:21.553
Elapsed time 11 minutes 0.54 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



****
Largest Lyapunov exponent calculation program by Andrew Dick
20-Aug-2003
2:45:32.296

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using First Zero of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89958 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.94517

5:53:16.322
Elapsed time 187 minutes 44.026 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****

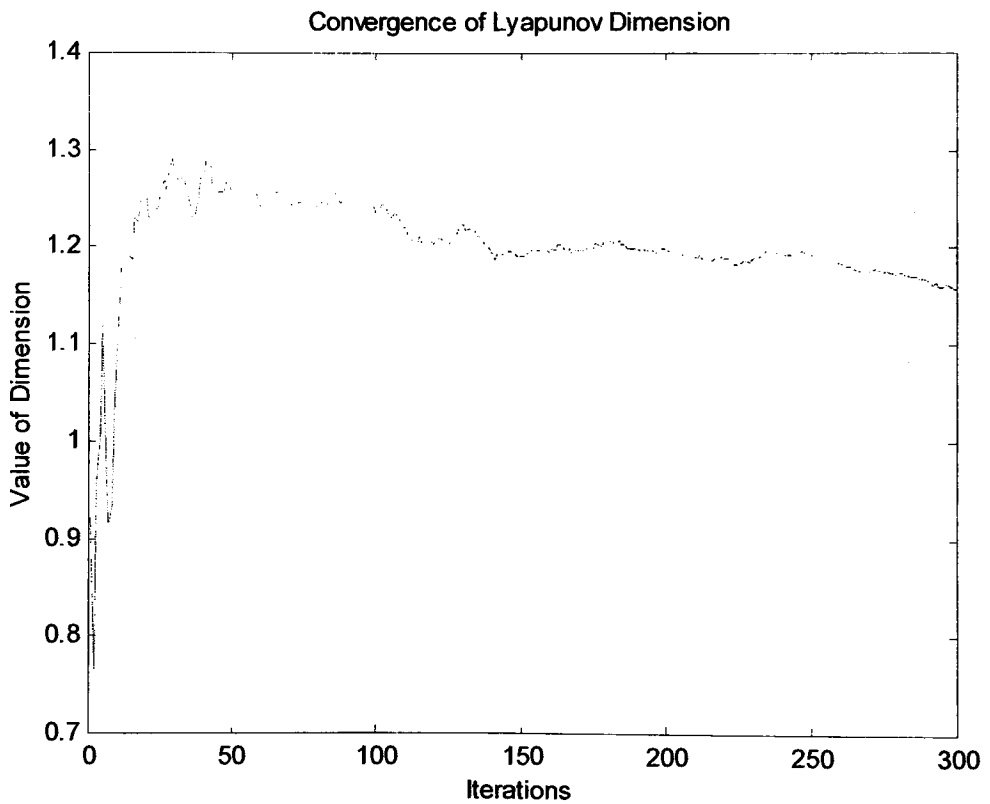


****

Largest Lyapunov exponent calculation program by Andrew Dick
20-Aug-2003
5:53:19.006

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using First Minimum of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89958 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 1.1578

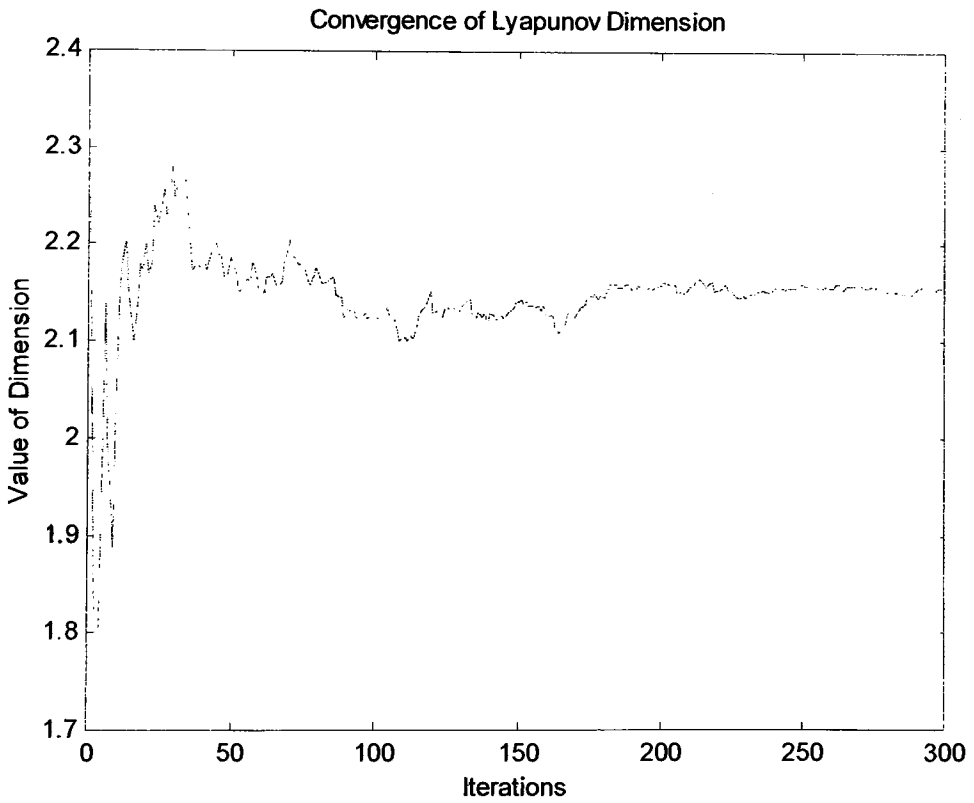
8:54:30.248
Elapsed time 181 minutes 11.242 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



****
Largest Lyapunov exponent calculation program by Andrew Dick
20-Aug-2003
8:54:32.942

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using First Inflection Point of Autocorrelation
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89958 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 2.1557

12:0:54.11
Elapsed time 186 minutes 21.168 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****

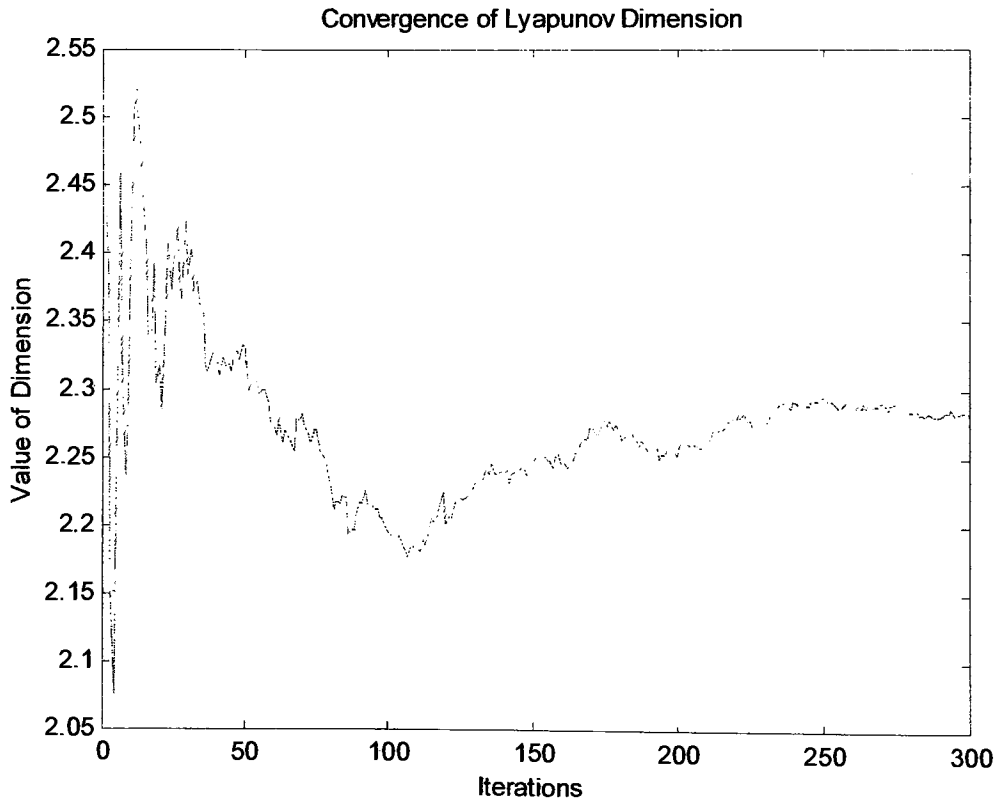


****

Largest Lyapunov exponent calculation program by Andrew Dick
20-Aug-2003
14:55:21.271

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using Average Displacement Method
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89970 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 2.2811

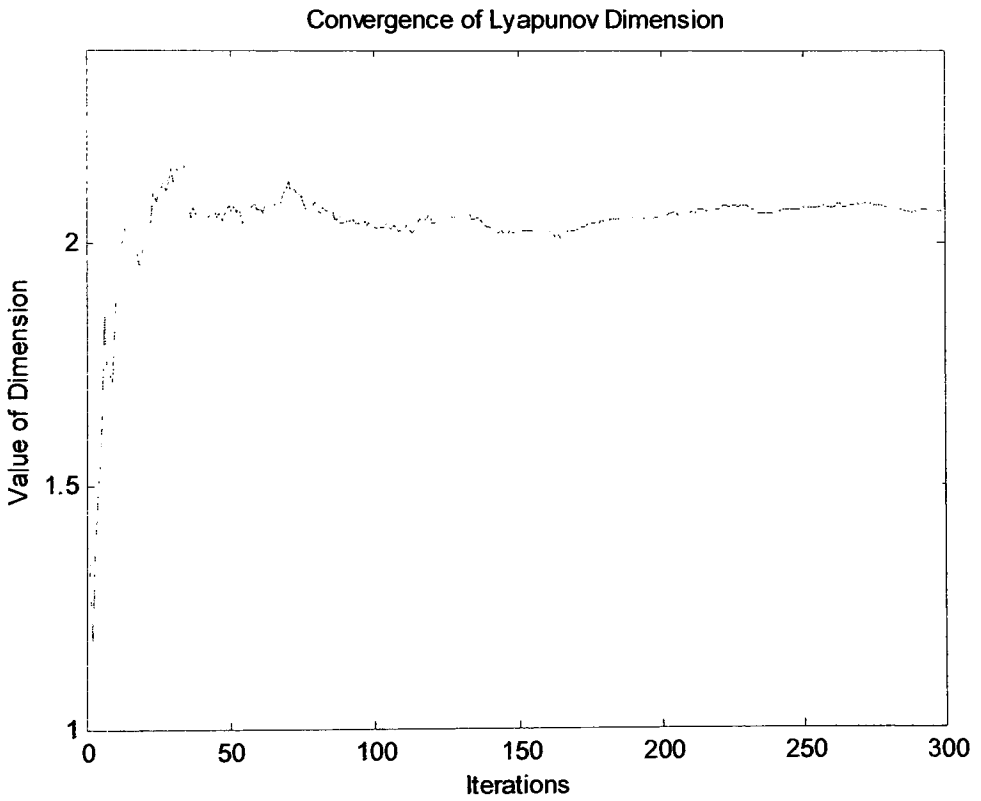
17:53:41.707
Elapsed time 178 minutes 20.446 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



****
Largest Lyapunov exponent calculation program by Andrew Dick
20-Aug-2003
17:53:44.391

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using Average Mutual Information Method
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89970 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 2.0689

20:46:55.783
Elapsed time 173 minutes 11.392 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



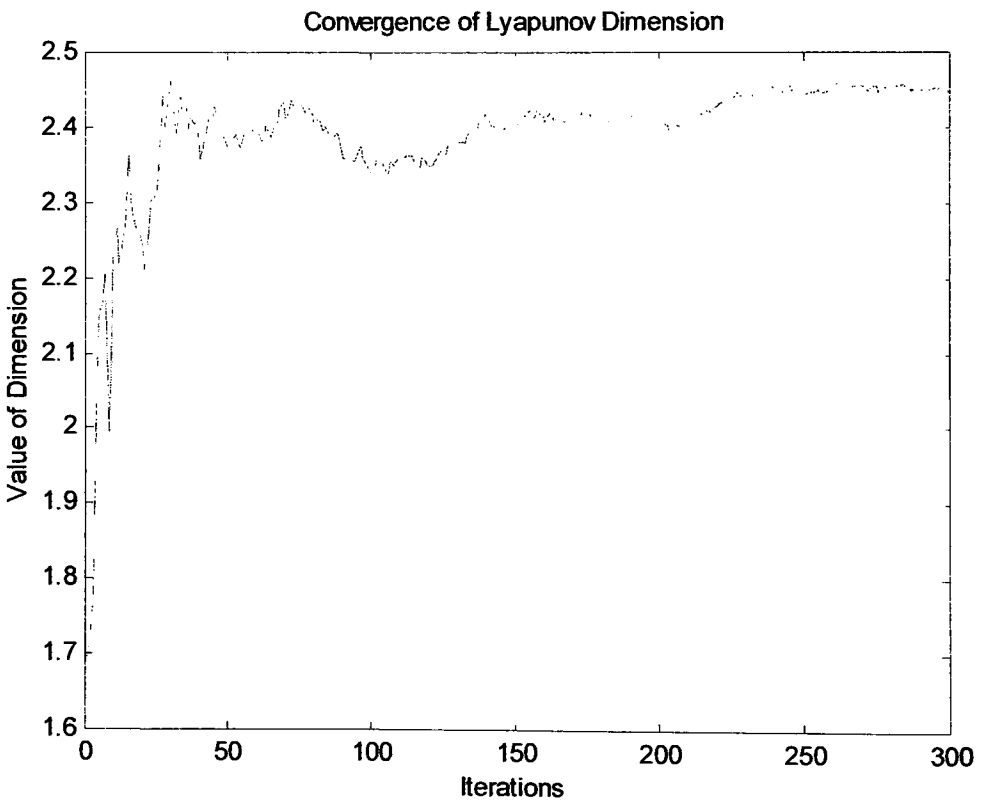
****

Largest Lyapunov exponent calculation program by Andrew Dick
26-Aug-2003
19:5:54.844

Data previously iterated using 5th order Lie Series approximation
Lorenz system with Popular Parameter Set, $\sigma = 16$, $r = 45.92$, $b = 4$
Reconstructed using Singular System Approach
and Embedding Dimension of 7
First 0.01 seconds of transient trajectory removed
Calculated from 89991 data points with time step of 0.01 seconds
Calculated using 300 2.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 2.4449

19:16:11.641
Elapsed time 10 minutes 16.807 seconds
Normal Termination of LARGEST_LYAPUNOV.M

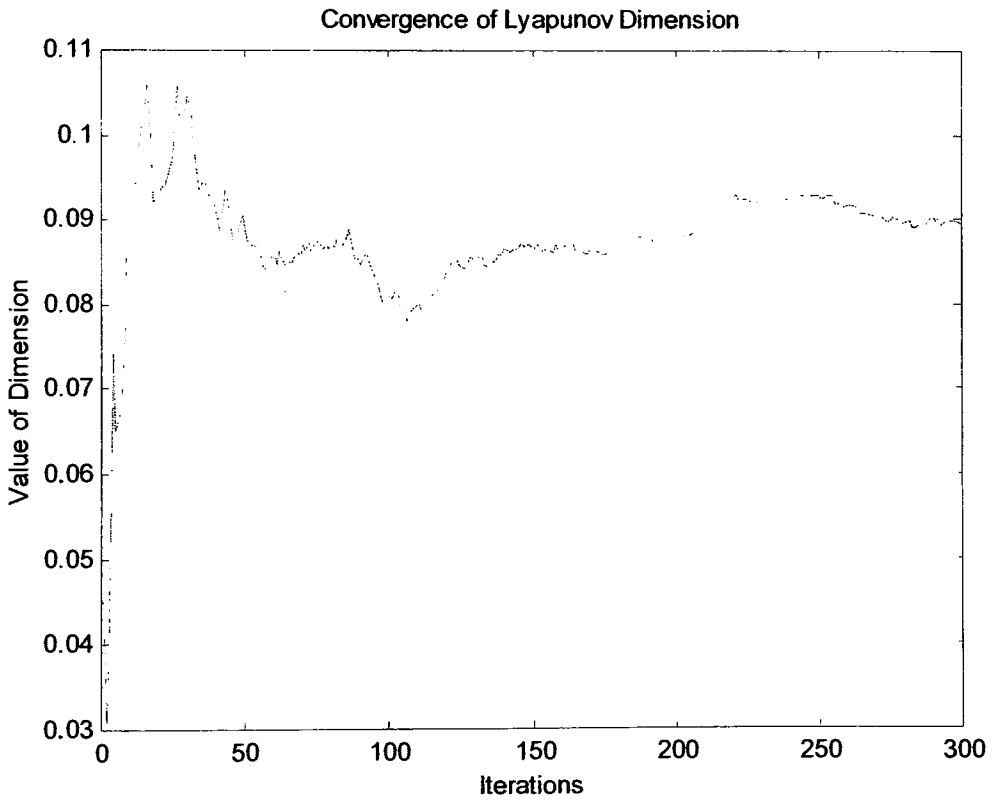
****



****
Largest Lyapunov exponent calculation program by Andrew Dick
21-Aug-2003
19:26:10.373

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using First Zero of Autocorrelation
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated from 89790 data points with time step of 0.05 seconds
Calculated using 300 12.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.089452

22:31:17.013
Elapsed time 185 minutes 6.64 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****

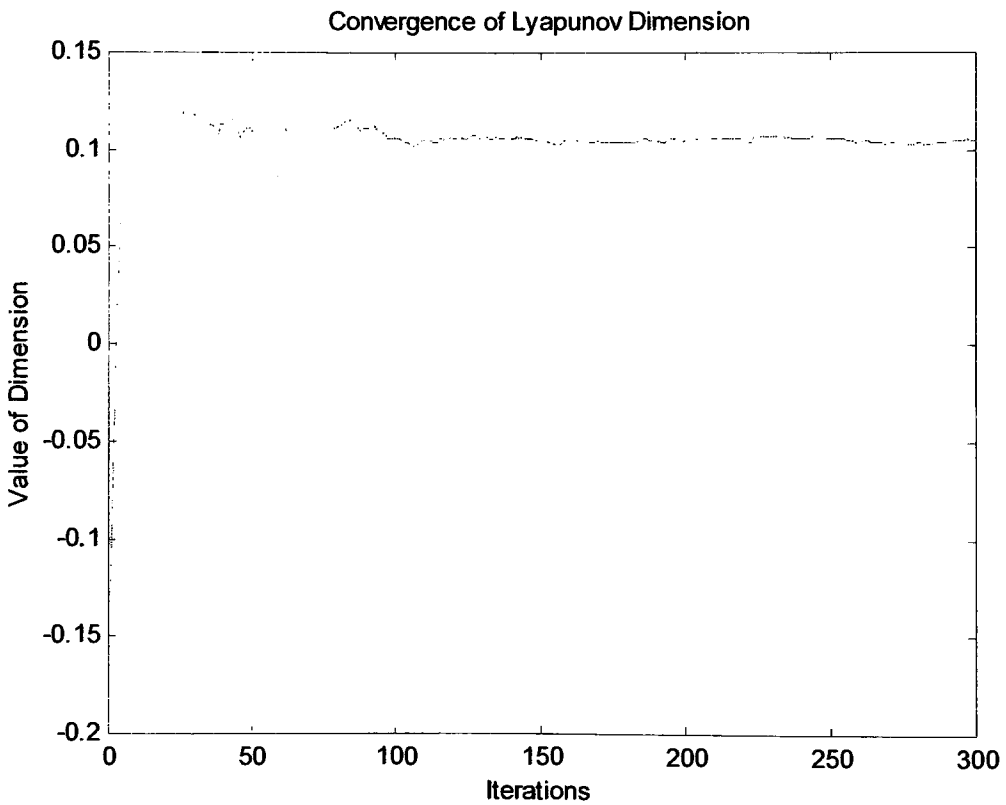


****

Largest Lyapunov exponent calculation program by Andrew Dick
21-Aug-2003
22:31:20.088

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using First Minimum of Autocorrelation
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated from 89790 data points with time step of 0.05 seconds
Calculated using 300 12.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.1049

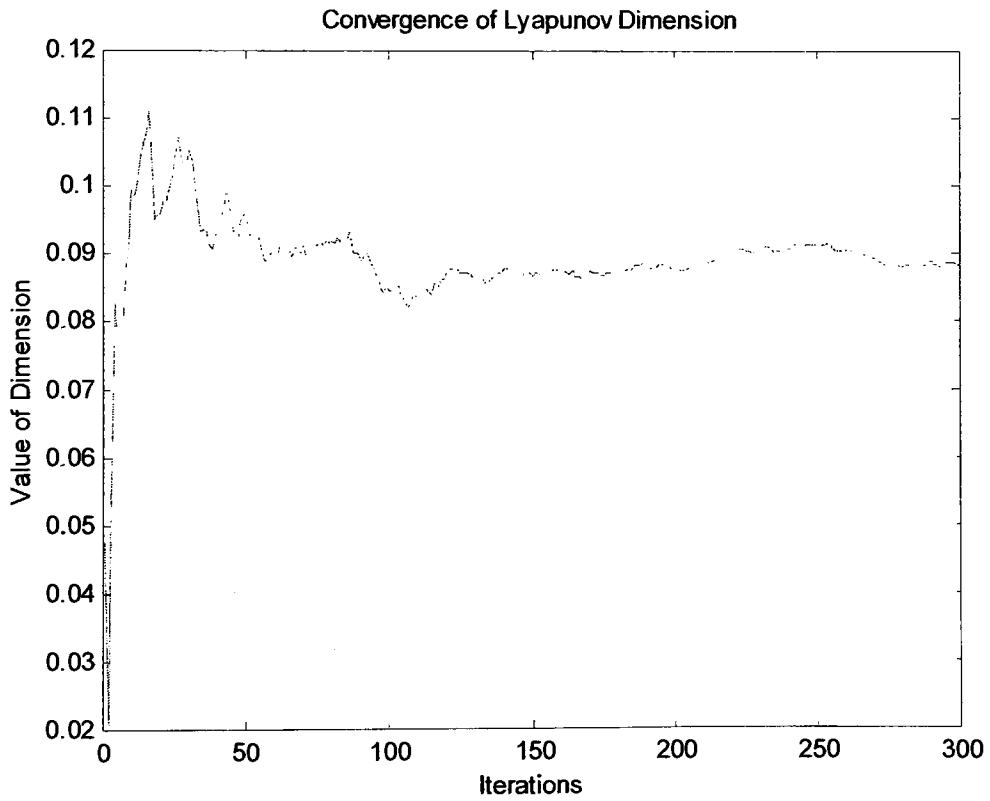
1:31:39.746
Elapsed time 180 minutes 19.658 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



****
Largest Lyapunov exponent calculation program by Andrew Dick
22-Aug-2003
1:31:43.221

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using First Inflection Point of Autocorrelation
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated from 89804 data points with time step of 0.05 seconds
Calculated using 300 12.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.088328

4:37:21.977
Elapsed time 185 minutes 38.766 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****

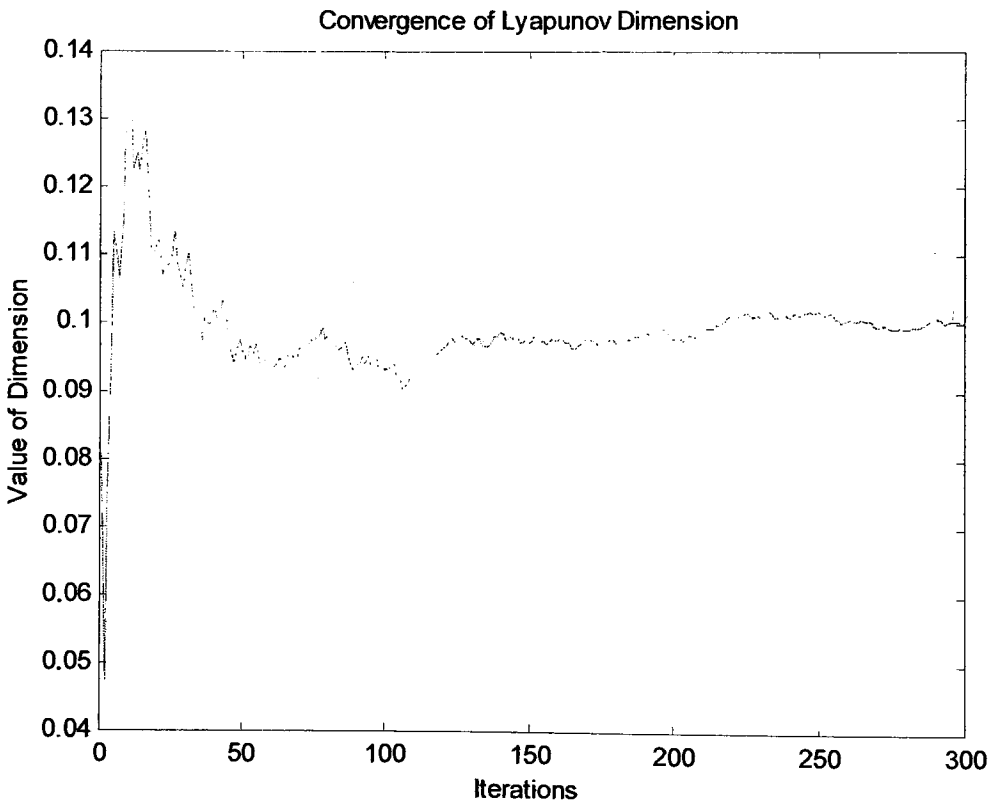


****

Largest Lyapunov exponent calculation program by Andrew Dick
22-Aug-2003
7:38:19.8

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using Average Displacement Method
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated from 89937 data points with time step of 0.05 seconds
Calculated using 300 12.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.10042

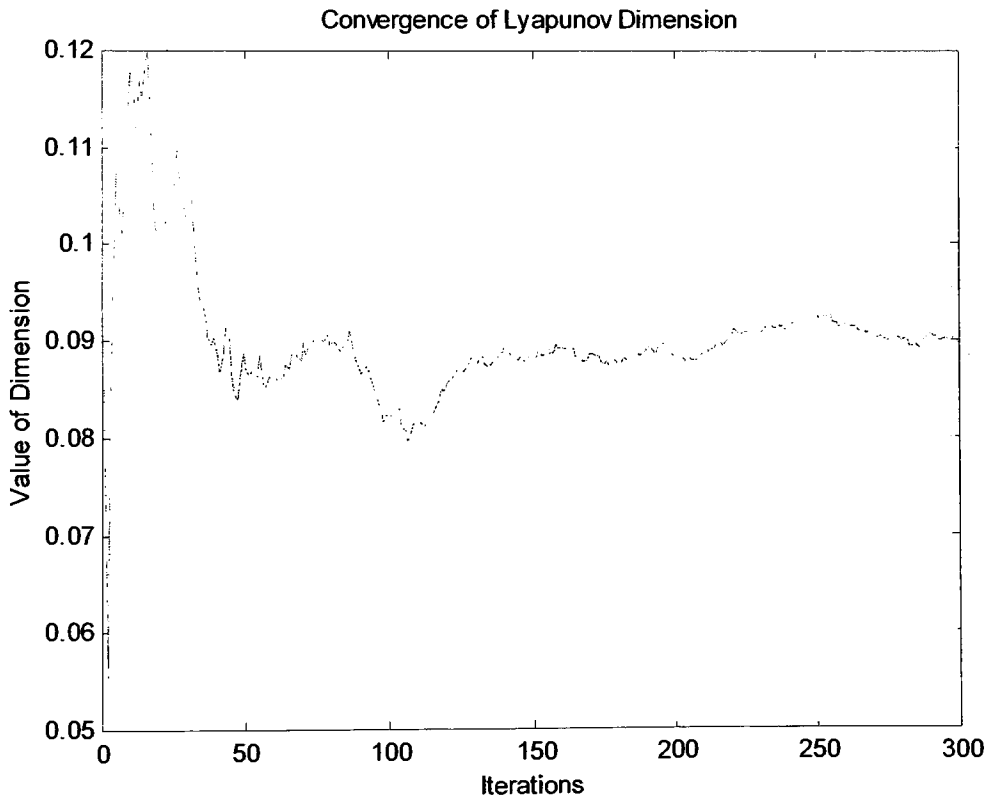
10:45:35.997
Elapsed time 187 minutes 16.197 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



****
Largest Lyapunov exponent calculation program by Andrew Dick
22-Aug-2003
10:45:39.141

Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using Average Mutual Information Method
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated from 89937 data points with time step of 0.05 seconds
Calculated using 300 12.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.089709

13:44:43.161
Elapsed time 179 minutes 4.02 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****

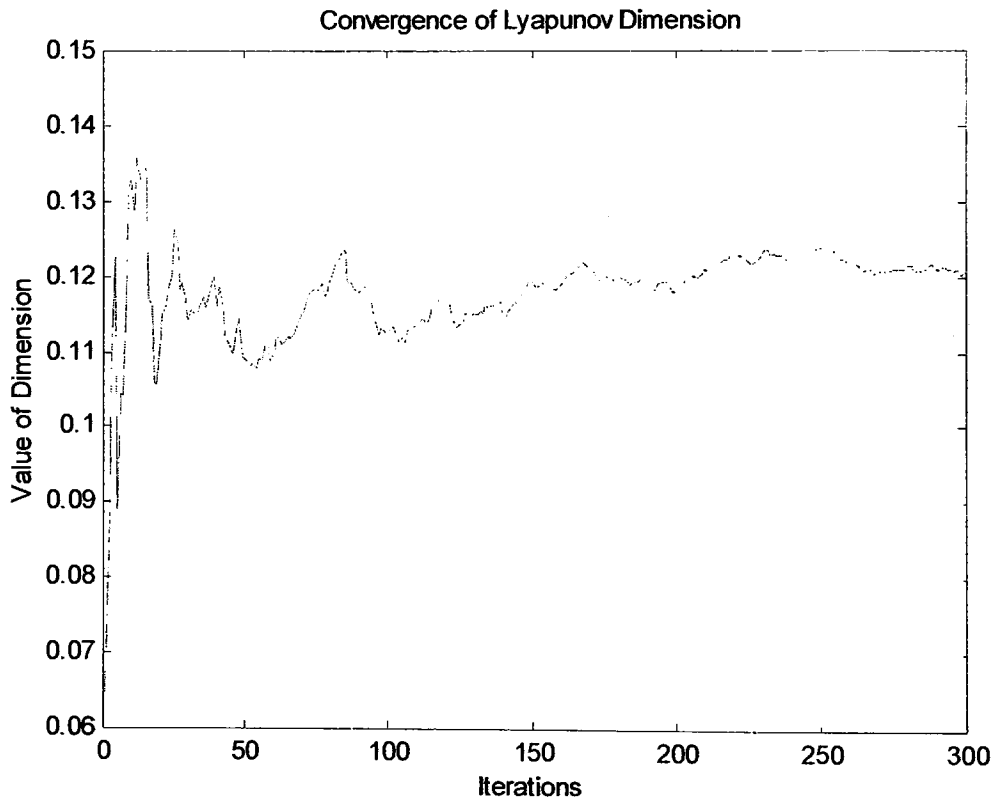


****

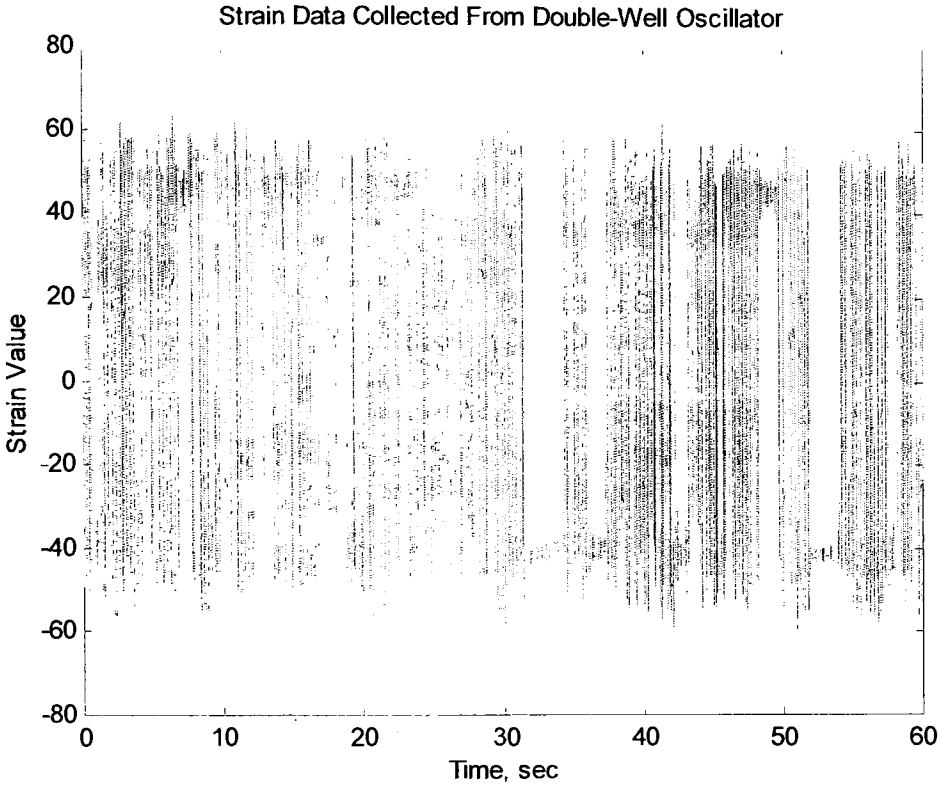
Largest Lyapunov exponent calculation program by Andrew Dick
26-Aug-2003
21:14:0.265

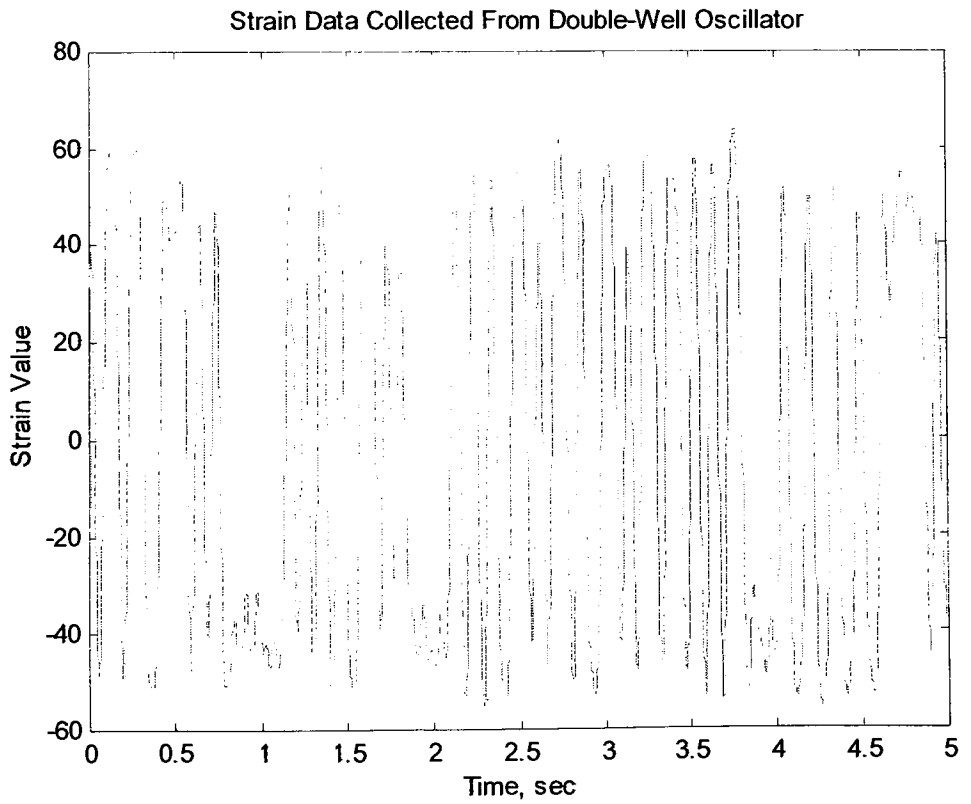
Data previously iterated using 5th order Lie Series approximation
Rössler system with Popular Parameter Set, $a = 0.15$, $b = 0.20$, $c = 10$
Reconstructed using Singular System Approach
and Embedding Dimension of 8
First 0.05 seconds of transient trajectory removed
Calculated from 89972 data points with time step of 0.05 seconds
Calculated using 300 12.5 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 0.12075

21:30:42.456
Elapsed time 16 minutes 42.191 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



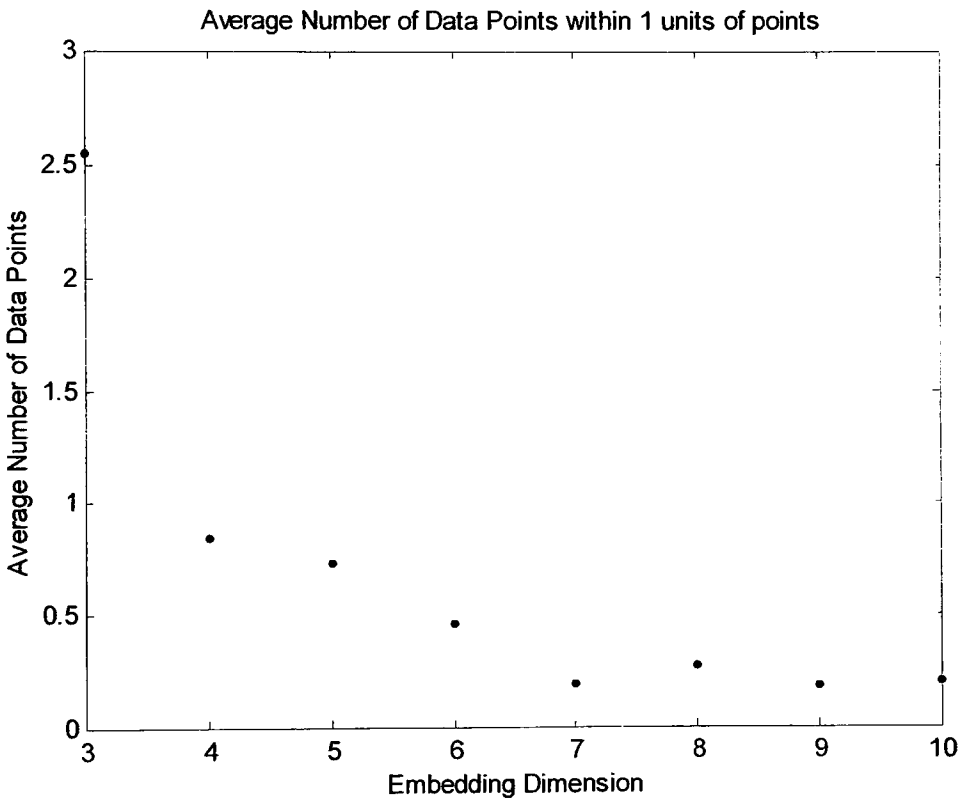
APPENDIX C13





```
**************************************************************************
Delay Time value calculation program using autocorrelation by Andrew Dick
27-Aug-2003
22:58:36.059

Data Collected From Double-Well Oscillator With Sampling Rate of 1000 Hz
Data set consisted of 5001 data points with a time step of 0.001 seconds
A transient period of 0 seconds was removed

Delay Value at First Zero is 0.173 seconds
Delay Value at First Local Minimum is 0.065 seconds
Delay Value at Half of Maximum Value is 0.032 seconds
Delay Value at 1/e of Maximum Value is 0.038 seconds
Delay Value at One-Tenth of Maximum Value is 0.056 seconds
Delay Value at First Inflection Point is 0.029 seconds

22:58:36.82
Elapsed time 0 minutes, 0.761 seconds
Normal Termination of Program AUTOCORRELATION_DELAY.M
**************************************************************************
```

```
**************************************************************************
Calculation of Embedding Dimension by Andrew Dick
27-Aug-2003
22:58:36.82

To remove a sufficient amount of false nearest neighbors,
  an embedding dimension of 7 is required

23:10:39.859
Elapsed time 12 minutes, 3.039 seconds
Normal Termination of Program EMBEDDING_DIMENSION.M
**************************************************************************
```



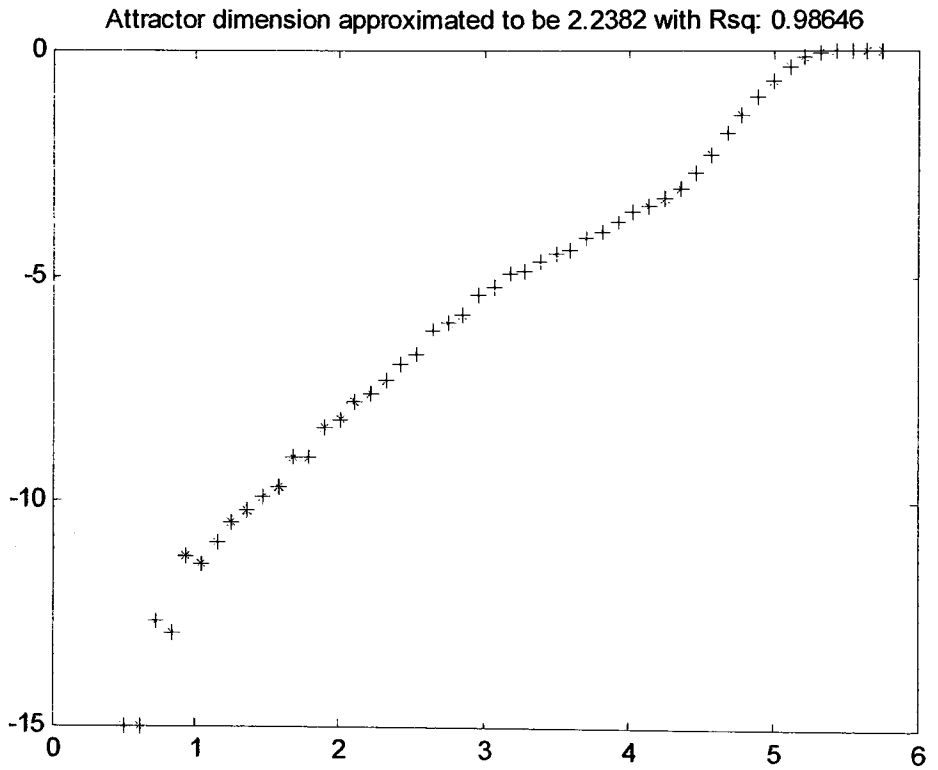
****

Correlation Dimension calculation program by Andrew Dick
01-Sep-2003
14:32:29.699

Data Collected From Double-Well Oscillator With Sampling Rate of 1000 Hz
Calculated using 5000 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 2.2382

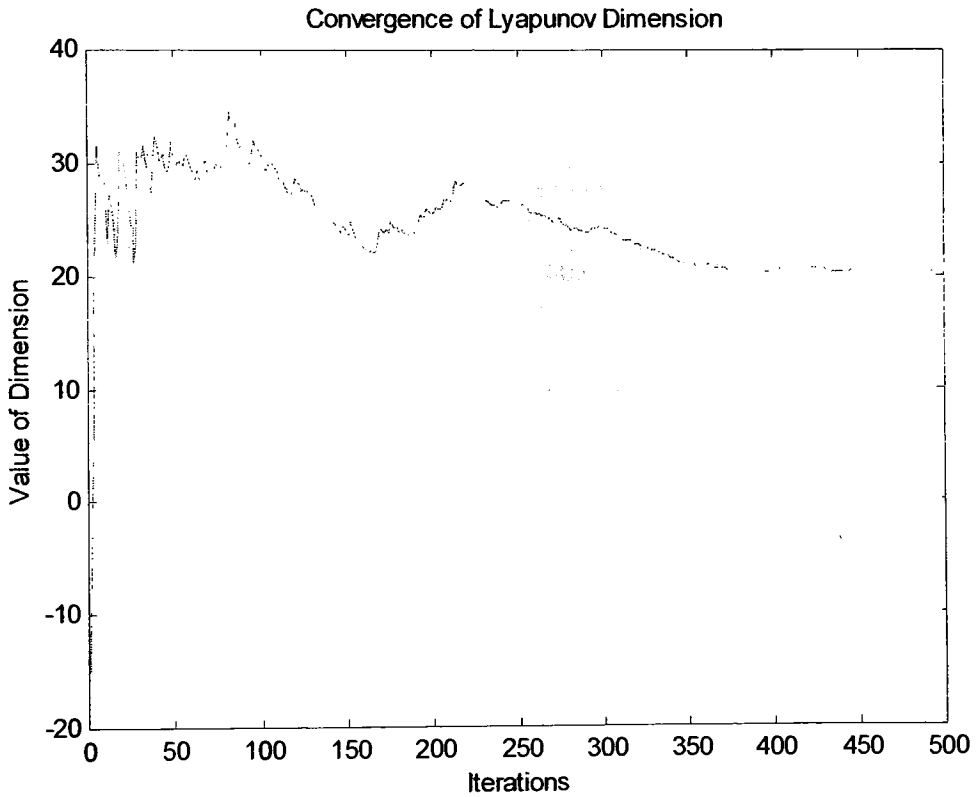
15:8:38.047
Elapsed time 36 minutes 8.348 seconds
Normal termination of CORRELATION_DIMENSION.M
****



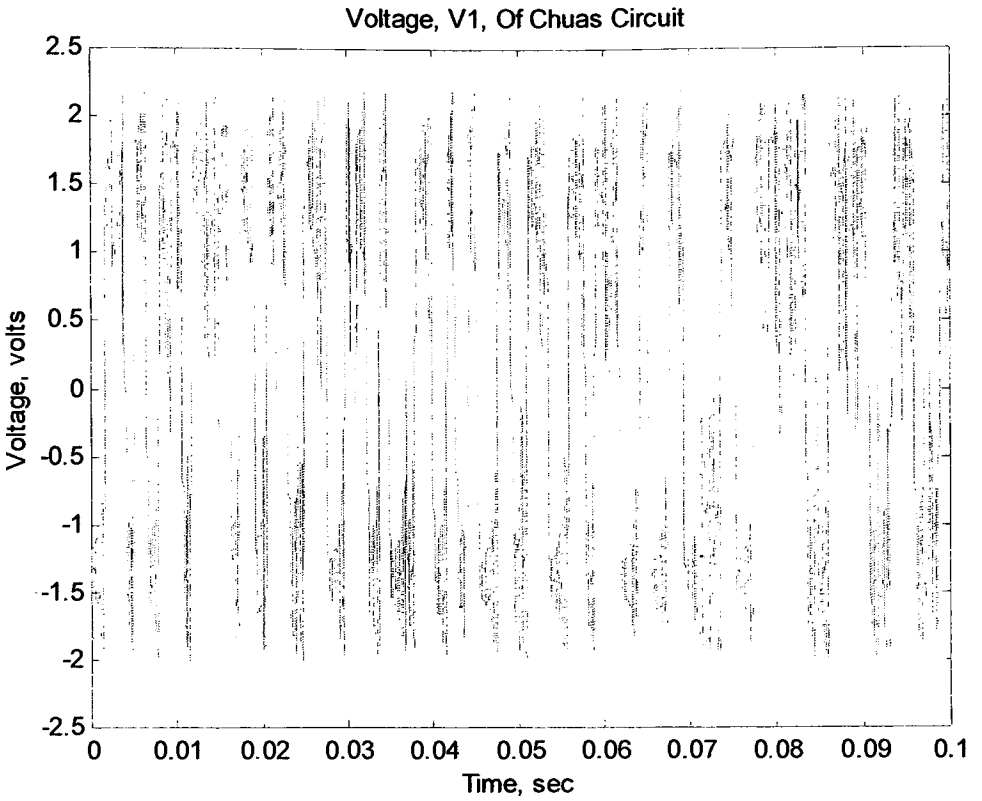
****
Largest Lyapunov exponent calculation program by Andrew Dick
01-Sep-2003
15:8:38.047

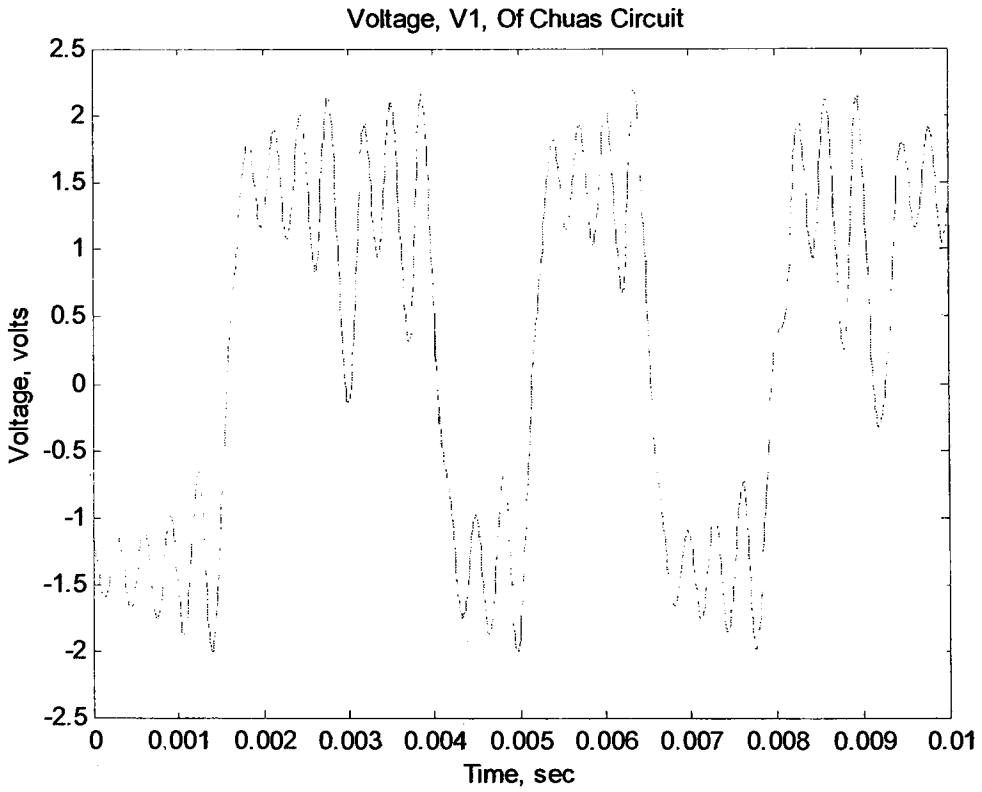
Data Collected From Double-Well Oscillator
Calculated from 59826 data points with time step of 0.001 seconds
Calculated using 500 0.01 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 20.0495

15:26:0.195
Elapsed time 17 minutes 22.148 seconds
Normal Termination of LARGEST_LYAPUNOV.M
****



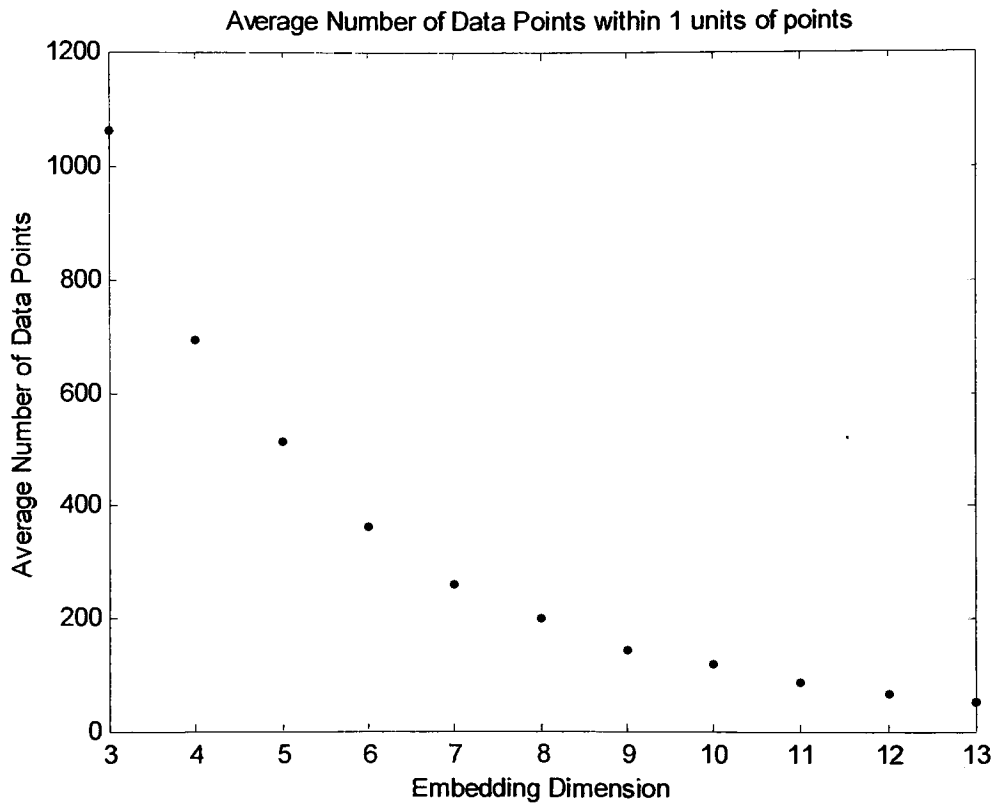
APPENDIX C14



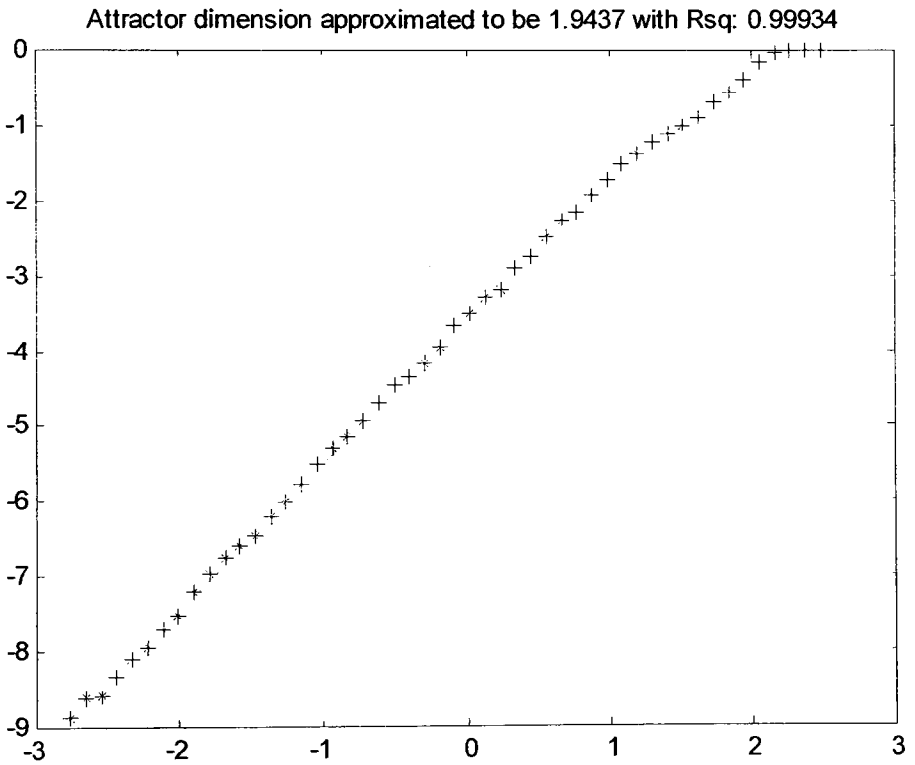


```
**************************************************************************
Delay Time value calculation program using autocorrelation by Andrew Dick
28-Aug-2003
7:36:24.553

Data Collected From Chua's Circuit
Data set consisted of 5001 data points with a time step of 1e-005 seconds
A transient period of 0 seconds was removed

Delay Value at First Zero is 0.0009 seconds
Delay Value at First Local Minimum is 0.00142 seconds
Delay Value at Half of Maximum Value is 0.00044 seconds
Delay Value at 1/e of Maximum Value is 0.00054 seconds
Delay Value at One-Tenth of Maximum Value is 0.00079 seconds
Delay Value at First Inflection Point is 0.0001 seconds

7:36:25.264
Elapsed time 0 minutes, 0.711 seconds
Normal Termination of Program AUTOCORRELATION_DELAY.M
**************************************************************************
```

```
**************************************************************************
Calculation of Embedding Dimension by Andrew Dick
28-Aug-2003
7:36:25.264

To remove a sufficient amount of false nearest neighbors,
  an embedding dimension of 9 is required

7:56:27.523
Elapsed time 20 minutes, 2.259 seconds
Normal Termination of Program EMBEDDING_DIMENSION.M
**************************************************************************
```



```
**************************************************************
Correlation Dimension calculation program by Andrew Dick
01-Sep-2003
11:1:9.766

Data Collected From Chua's Circuit With Sampling Rate of 10000 Hz
Calculated using 5000 data points
Calculated using a Theiler coefficient of W=10

The Correlation Dimension of the data set was determined to be 1.9437

12:35:45.287
Elapsed time 94 minutes 35.551 seconds
Normal termination of CORRELATION_DIMENSION.M
**************************************************************
```



****

Largest Lyapunov exponent calculation program by Andrew Dick
01-Sep-2003
13:49:32.613

Data previously iterated using 5th order Lie Series approximation
First 0 seconds of transient trajectory removed
Calculated from 9920 data points with time step of 1e-005 seconds
Calculated using 800 0.0001 seconds segments
The largest Lyapunov exponent for this data set is calculated to be 1558.9256

13:53:7.642
Elapsed time 3 minutes 35.029 seconds
Normal Termination of LARGEST_LYAPUNOV.M

****